Supporting the Construction of Real World Interfaces

D. Scott McCrickard, David Wrighton, Dillon Bussert
Department of Computer Science
Virginia Polytechnic Institute and State University
{mccricks, dwrighto, dbussert}@vt.edu

Abstract

In recent years, real world objects have been used to reflect information previously shown on the computer screen. While most earlier efforts have required significant developer knowledge and skills to construct and program the displays, our work is investigating methods to enable programmers to use real world objects in much the same way that they would typical user interface widgets. This paper outlines programming interfaces developed on top of the X10 protocol for controlling power flow to electrical devices, including sample programs for each.

1 Overview

With the availability of information sources like the World Wide Web, users often need to stay aware of constantly changing information, or at least have the information readily available at appropriate times and in appropriate forms. Quite often, the information is not so important that it should occupy space in a display on the computer desktop, yet it is of sufficient interest that users want it readily available. One solution that has become common in recent years has been to incorporate the information in the environment, an approach termed *ubiquitous computing* or *pervasive computing*. In these fields, non-traditional objects in the environment are used to reflect information of interest; for example, changes in lighting can correspond to the weather forecast, changes in air flow could reflect fluctuations in the stock market, and changes in ambient music could signal an upcoming meeting [5, 3, 2]. A nearby person could look at or sense the change without devoting significant attention to it and certainly without having a display that occupies valuable computer screen real estate.

These and other devices have been constructed using real world objects to reflect information, but typically they are not easy to construct. This goal of the work described in this paper is to provide methods and programming interfaces that support the creation of information displays using real-world objects. We call the resulting informational displays real-world interfaces, or RWIs. Typical user interfaces use buttons, scrollbars, sliders, and similar on-screen visual displays to convey and interact with information, whereas RWIs augment or replace these displays with changes in the surrounding environment that convey information in a less direct but still noticeable manner.

Perhaps the effort most similar to our own comes from Saul Greenberg and his phidgets project [1]. Phidgets, short for physical widgets, provide a layer of abstraction between physical devices and the programmer, allowing programmers to focus on the creation of the physical device and its inclusion in a software package. However, phidgets still require specially adapted hardware, and most of the phidget applications required a fair amount of effort to be spent in the physical construction of the interaction device. Our goal is to provide a programming interface that minimizes or eliminates the need for electronics expertise.

Our work provides programmers with the ability to use real-world devices in much the same way that they would use a standard user interface toolkit. We have developed the Real World Interface (RWI) library, a protocol and set of abstract programming interfaces (APIs) for use in controlling X10 devices. The RWI library empowers programmers with the ability to create their own RWIs. A programmer can use a familiar API in displaying information using any X10 device.

The X10 protocol traditionally has been used for home automation. The protocol is most commonly found in devices that control the flow of electricity to appliances by toggling power or setting power levels. Other devices that have been built using the X10 protocol include motion detectors, cameras, thermostats, and remote controls. Signals can be sent from a computer over power lines to X10 devices within a limited range like a house or a lab.

To support multiple programming paradigms, and to understand how different paradigms are applicable in the concept of X10 devices, the RWI library supports multiple languages and toolkits. We attempted to model the APIs on paradigms used when programming in that language or

toolkit. With compact, simple commands using a syntax familiar to programmers, a program can be written to control the power levels to electrical devices. The end product is a library that, with appropriate off-the-shelf hardware, allows programmers to create applications for real-world devices.

We have used the RWI library in the development of several applications for home and small office use. Since real world devices reflect information in a way not necessarily obvious to outsiders, it seems that individuals and smaller communities of users, where the meaning of changes in the environment can be established more easily, would benefit most from RWIs. In turn, RWIs could help increase the sense of community by generating conversation about information they represent, including anything from the state of machines in a lab to the current weather conditions.

This paper describes the difficulties in creating RWIs using traditional means, and introduces our RWI library that addresses these difficulties. We also describe several applications built using the RWI library. Note that this paper does not provide the full syntax and description for most commands in the APIs; please visit the RWI Web site at http://www.cs.vt.edu/rwi for additional information and access to the library.

2 Bridging the gap to the real world

Our initial concern was to make it practical to construct real world interfaces. The limitations inherent to real world devices (and particularly X10 devices) require significant programming to overcome, and some researchers feel that X10 in particular is simply too high-level and limited for general device development [1]. However, we have found that shielding the programmer from these issues with threading and error handling techniques makes it possible to use these devices at an acceptable level of performance.

To avoid blocking and to better support error checking, we developed a queue-based multithreaded layer based on the concept of passing message objects to virtual device objects that then control the X10 commands. Messages intended for X10 devices are entered into the input queue maintained in order by timestamp. A helper thread dequeues the messages from the input queue and enqueues them in the appropriate device queue. Messages are dequeued from the device queues in circular order at the appropriate time intervals to limit blocking and failure often found with X10 devices. The messages are sent to a virtual device, which sends the message to the actual X10 device and reports success or failure.

This layer supplements the standard X10 command set to make it possible to write programs that use real world objects to reflect information. Several APIs not only make it possible but furthermore potentially simplify the job of the programmer. The APIs provide abstractions that hide many of the details of the C++ multithreaded layer using familiar programming paradigms for several languages and toolkits: C++, C, Amulet, and Tcl/Tk.

The C++ interface extends the C++ multithreaded layer with commands that support easy access to the functionality. The C interface exposes a procedural interface and the use of an opaque handle to maintain state. The Amulet interface creates an Amulet object class which can take advantage of constraints and command objects common to widgets in the Amulet system [4]. The interface for Tcl/Tk supports rapid development of interfaces with the Tcl language whereby real world objects are manipulated similar to graphical objects in the Tk toolkit. Complete documentation is available at http://www.cs.vt.edu/rwi/

3 RWI applications

Limitations inherent to real world devices and to X10 in particular constrain the types of applications that can be designed for the RWI library. Specifically, the low information transfer rate makes displays that change frequently difficult or impossible to create. One often suggested real world interface is an intrusive display that rapidly flashes a light or changes some other visual cue to alert someone of a change in information of interest. While the low transfer rate makes this sort of display impossible, we also argue that this interface is undesirable. If the goal is to grab a person's attention because some urgent event is taking place, a visual desktop alarm or audio warning seems ideal. If, on the other hand, the goal is to make information available, using the environment to reflect the state of information seems appropriate. When users care deeply about some specific change, they are more tolerant of intrusive interfaces that pop up on their computer screens or create an audio alert.

We share the vision of Mark Weiser that interfaces integrated into the environment should encalm, not intrude [5]. That is, the display of information should blend into the environment with minimal negative impact upon other tasks, yet it should be visible at the times when a user wants or needs it. This section describes several applications developed as a means of demonstrating both the functionality of the RWI library and the areas where we feel real world interfaces are most appropriate. For other examples, please visit the RWI Web site at http://www.cs.vt.edu/rwi/

Alarm clock Perhaps the simplest application we developed was a 10-line alarm clock program that dimmed a desk lamp as a meeting time approached. To use the application, a user types a command `alarm 60` to set a RWI alarm for 60 minutes from the current time. As the meeting time approaches, the lamp dims, providing a subtle reminder of the approaching meeting. When the meeting time arrives, the lamp turns off completely, providing a more noticeable meeting alert.

Figure 1. The weather monitor RWI. The brightness of the lamp reflects the current temperature, and the fan reflects whether it is currently raining. The RWIs remind people to consider bringing a jacket or umbrella.

Weather monitor The Web provides abundant sources of changing information that could be reflected using RWIs. As our lab has no windows to the outside world, one important information source of interest is the current weather. All too often someone would leave the building only to discover that it was raining or the temperature had dropped significantly. To alleviate this problem, we created a RWI that monitored the weather and used a lamp and a fan to reflect the weather at the time (see Figure 1). We situated the displays by the door of the lab, as we expected people would most need the information when leaving.

Computer use monitor Often it is important to maintain a sense of how much time has been spent using a computer, device, or application. For example, the RWI project group does most of its development on a single machine in our lab, and it is often interesting to speculate when the machine has been in use. We constructed a RWI using an electric clock that we reset to 12:00 at each lab meeting. Upon login to the RWI account from the console, the clock begins to run, and upon logout it stops. At each meeting, we can tell how much time has been spent recently on development.

An important and straightforward alternative to this application would be a keyboard usage monitor. By setting the clock to 12:00 each day and configuring it to run only when the keyboard has been recently used, it could continually reflect how much time has been spent using the keyboard during a day. We see this as highly useful for repetitive stress injury (RSI) sufferers who need to limit the time that they spend typing. The clock would provide a constant but off-the-desktop reminder of how much time has been spent during the day typing and could help an RSI sufferer limit typing and manage breaks.

4 Conclusions and future work

The work described in this paper outlines a method for controlling real world devices using APIs similar to those found in widely used languages and toolkits. Our RWI library shields a programmer from the speed and reliability problems that are common when using real world devices, particularly those that make use of the X10 protocol. In addition, our APIs simplify the job of the programmer in creating interfaces that leverage objects in the real world. Applications were described that have been implemented using the RWI library.

While this and other research efforts have addressed issues related to the representation of information using real world objects, the research area is still in its infancy with many problems remaining to be solved. An initial step will be to extend the framework to include more interactive devices like motion detectors, cameras, and remote controls, all available as X10 devices. Since our ultimate goal is to make it easier not just for programmers but for the average user to construct RWIs, future work should also continue to add framework layers that enable users with little programming or even computer experience to build and use RWIs.

Finally, it is necessary to understand the effectiveness of real world interfaces through empirical testing. While RWIs we and others have constructed have been anecdotally described as useful and entertaining, empirical testing is needed to determine the usefulness and appropriateness of RWIs in a variety of situations. We suspect that there are tradeoffs between prompting reactions, supporting comprehension and interrupting current tasks. By identifying key aspects of these tradeoffs, we can understand better how real world interfaces can and should be constructed and used.

References

[1] S. Greenberg and C. Fitchett. Phidgets: Easy development of physical interfaces through physical widgets. In *Proceedings of the ACM Conference on User Interface Software and Technology (UIST '01)*, Orlando, FL, Nov. 2001.

[2] D. Gruen, S. Rohall, N. Petigara, and D. Lam. "in your space" displays for casual awareness. In *Demonstrations at the ACM Conference on Computer Supported Collaborative Work (CSCW '00)*, 2000.

[3] H. Ishii and B. Ulmer. Tangible bits: Towards seamless interfaces between people, bits, and atoms. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '97)*, pages 234–241, Atlanta, GA, Mar. 1997.

[4] B. A. Myers, R. G. McDaniel, R. C. Miller, A. S. Ferrency, A. Faulring, B. D. Kyle, A. Mickish, A. Klimovitski, and P. Doane. The amulet environment: New models for effective user interface software. *IEEE Transactions on Software Engineering*, 23(6):347–365, June 1997.

[5] M. Weiser and J. S. Brown. Designing calm technology. *PowerGrid Journal*, 1.01, July 1996.