A Framework for using the iPhone as a Wireless Input Device for Interactive Systems

Jonathan Diehl
Media Computing Group
RWTH Aachen University
diehl(at)cs.rwth-aachen.de

Jan-Peter Krämer
Media Computing Group
RWTH Aachen University
jpk(at)cs.rwth-aachen.de

Jan Borchers
Media Computing Group
RWTH Aachen University
borchers(at)cs.rwth-aachen.de

ABSTRACT
We have developed a framework to allow quick prototyping of the input channels provided by the iPhone, namely multi-touch and accelerometer data, for interactive systems. The framework consists of two parts: a native iPhone application captures and forwards events to the client application, which is built utilizing our framework. We hope to encourage further research into the use of innovative input methods by taking some of the implementation effort away from the researcher.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. – Input Devices.

General terms: Design

Keywords: Input device, iPhone, Multi-Touch, Accelerometer

INTRODUCTION
Much research effort goes into exploring new input devices and finding appropriate use patterns for them. However, the cost of creating input devices, which offer these new capabilities, and coupling them with interactive systems to test them, is often very high.

The iPhone offers two innovative input channels, accelerometer data and multi-touch capabilities, conveniently bundled into a mobile device. Many researchers are currently working on these channels, Johnny Lee's work with the Wiimote and Jeff Han's work on multi-touch probably being the most prominent projects.

To allow widespread use of these metaphors, we have created a framework supporting easy integration of these input channels into interactive systems. As a next step we want to supplement our framework with the capability to render a custom user interface on the iPhone for controlling the interactive system. We hope to achieve this before the publication of this poster.

RELATED WORK
There are many solutions for bringing physical interfaces closer to the computer. [3] introduced Phicons (physical icons) to allow tangible interaction with the computer. [2] presented the more general toolkit of physical user interface elements called Phidgets (physical widgets). The Calder Toolkit [4] allows quick prototyping of hardware input devices based on microcontrollers. All these toolkits allow researchers to integrate a range of low- and high-level sensors and actuators into their designs. Our framework can be seen as a supplement, which brings multi-touch and detailed accelerometer data to the researchers hands.

The iStuff Toolkit [1], on the other hand, supports researchers in integrating these hardware prototypes into an interactive environment. For the iPhone the integration into the iStuff environment is possible without a proxy by making the device connect directly to the service.

There are several solutions to control a desktop computer remotely using the iPhone. These solutions however suffer from high latency due to high network traffic and do not support multi-touch or accelerometer events.

Johnny Lee has explored the use of Wiimotes as input devices for desktop computers. They support a 3D accelerometer similar to the iPhone and infrared tracking capabilities. There are several frameworks available for using the Wiimote with custom software. Nevertheless, we believe the iPhone's different form factor and its multi-touch capabilities offer a valuable alternative to the Wiimote.

APPROACH
Our work consists of two separate pieces of software, one running on the iPhone as a native application, and the other running on the desktop computer.

The iPhone server software captures input events and accelerometer data and prepares it to be sent over its wireless network interface to a desktop client.

A client registers with the iPhone application via IP or Bonjour, which allows effortless coupling inside a local area network. Afterwards, the client receives all events that are captured on the iPhone and provides them to the researcher.

Using such a lightweight method, allows very short event response times. Quick response times are especially important for input devices, as otherwise the interaction itself would suffer and test results would be biased.

UIST'08, October 19-22, 2008, Monterey, California, USA
ACM 978-59593-975-3/08/10.

TECHNICAL OVERVIEW

To set up the iPhone, it is sufficient to install the server application on the phone. Building a specialized version of the server becomes interesting when a custom user interface is desired. We discuss this in the Outlook section.

The desktop application is implemented by setting up an instance of the ITTouchClient class, provided by our framework. This instance manages the connection to the iPhone and receives its events, which are forwarded to a delegate, which can be any controller that implements the ITTouchClientDelegate protocol. The delegate paradigm allows developers to use arbitrary controllers to react to system events, with very little need for configuration. It is the preferred method for event handling in the Mac OS X operating system.

Delegate methods are triggered on connecting and disconnecting an iPhone to the server, on updating the accelerometer data, and for each touch event. These events are wrapped in ITEvent objects, containing ITTouch objects for each individual touch. An ITTouch object corresponds to the UITouch class from the iPhone OS, containing a timestamp, tap count, the actual and previous position, and a phase (began, moving, stationary, ended, or canceled). The canceled-phase is most likely triggered by a disruption of the interaction, for instance due to an incoming call.

Accelerometer data is encapsulated in ITAcceleration objects, which have three parameters (x, y, z) to represent the three axes. Accelerometer data is updated with 30Hz, which should be sufficient for almost any purpose. The refresh rate is limited by the hardware capabilities of the iPhone.

Bonjour service discovery is realized using the Bonjour library, available for any operating service. We provide an ITTouchBrowser class, which informs its delegate of newly found touch servers, running on an iPhone. Connecting to the server is then done via IP, leaving the Bonjour service optional but recommended.

The network communication is handled via TCP sockets and abstracted by the framework. To ensure high performance, we have implemented our own simple network protocol, which encodes events as ASCII strings. The protocol is kept flexible to support other information in the future, like custom events or user interface changes.

OUTLOOK

We hope that researchers will take our framework and design new interaction techniques for interactive systems and effectively explore the new design space the iPhone provides. Possible applications could be: new input techniques for the desktop computer, interacting with large public displays, computer-supported collaborative work, or remote control.

While we believe many useful applications can already be created using this framework, there are still some clear limitations, we want to eliminate before the publication of this poster.

First, the iPhone should be capable of rendering a custom user interface, which can create and send custom events to the desktop system. For instance, the multi-touch area could be segmented into multiple areas, which create individual events when touched. Further, the user interface should be extendible by standard user interface widgets, like sliders.

Besides visualizing input capabilities, we want to add support for feedback to the user. This feedback could be automatic, e.g. by visualizing touch events by briefly lighting up the area on the screen, or triggered by the interactive system. These changes could be direct, such that the interactive systems send the information to be displayed, or indirect using a state machine. The iPhone brings a high-resolution display, a speaker, and a vibration unit to allow feedback.

On a different note, it would be interesting to allow communication of multiple iPhones with multiple desktop systems. For instance, one iPhone could be used to control many large public displays at once, where the events are mapped differently for each screen. Similarly, many iPhones could be mapped to one desktop system to allow multi-player capabilities and collaborative work.

Consequently, the suggested coupling mechanism of using Bonjour on the desktop client side, might not be feasible for situations of multiple, loose clients. Here, more flexible coupling techniques must be employed, like scanning a barcode or location-tracking techniques.

REFERENCES

1. Ballagas, R., Ringel, M., Stone, M., and Borchers, J.. *iStuff: a physical user interface toolkit for ubiquitous computing environments*. In CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems, 2003, 537--544.
2. Lee, J. C., Hudson, S. E., Summet, J. W., and Dietz, P. H.. *Moveable interactive projected displays using projector based tracking*. In UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology, 2005, 63--72.
3. Greenberg, S. and Fitchett, C.. *Phidgets: easy development of physical interfaces through physical widgets*. In UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology, 2001, 209--218.
4. Ishii, H. and Ullmer, B.. *Tangible bits: towards seamless interfaces between people, bits and atoms*. In CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, 1997, 234--241.
5. Lee, J. C., Avrahami, D., Hudson, S. E., Forlizzi, J., Dietz, P. H., and Leigh, D.. *The calder toolkit: wired and wireless components for rapidly prototyping interactive devices*. In DIS '04: Proceedings of the 5th conference on Designing interactive systems, 2004, 167--175.