Testing Physical Computing Prototypes Through Time-Shifted & Simulated Input Traces

Timothy Cardenas, Marcello Bastea-Forte, Antonio Ricciardi, Bjoern Hartmann, Scott R. Klemmer
Stanford University HCI Group
Computer Science Department, Stanford, CA 94305
{trcarden | graphite | aricciardi | bjoern | srk}@cs.stanford.edu

ABSTRACT
To refine off-the-desktop user experiences, testing of interaction logic is essential. Experimenting with such interactions is harder than with normal GUI code as real-time input data may be time-intensive to obtain or may not be available at all. In addition, recognition-based interactions need to be tested on many cases to ensure robustness. To address these challenges a testing toolkit should satisfy four criteria: record and replay input event traces, simulate input data to create fake event traces in real-time, edit recorded data offline, and visualize/monitor input data that is presented to an application. We are constructing *FauxPut*, a testing toolkit for interaction designers that unifies these strategies in a common interface.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: input wrapper, simulated inputs

INTRODUCTION
As computing moves off the desktop and encompasses a wider range of sensors and actuators, the testing of sensor-aware applications becomes increasingly important. Unfortunately, generating useful test input data for novel interactions is often challenging. For example, providing live GPS input for a location-driven application requires leaving one's desk. Furthermore, testing specific edge-cases or inputs that go beyond the physical capabilities of the tester (e.g., testing multi-user interactions) is very difficult. We argue that an input toolkit specifically aimed at testing sensor-based interaction logic is needed to address these challenges.

This paper describes our ongoing effort of building *FauxPut*, a toolkit that allows designers to both time-shift and simulate traces of input event data. *FauxPut* transparently wraps input device APIs and provides an interface for recording, simulating and editing inputs for recognition-based interactions. *FauxPut* provides visual feedback of input traces and enables users to manually override and play back the data before it reaches the software being tested. This allows designers to quickly debug sensor-based applications without manually re-generating input data.

SCENARIO
We demonstrate the utility of the *FauxPut* system by prototyping a hands-free 3D navigation system (Figure 1).

UIST'08, October 19–22, 2008, Monterey, California, USA
ACM 978-1-59593-975-3/08/10.

Suppose a designer wants to base the position of the virtual camera in a 3D world on the X/Y position of the viewer's head with respect to the display (through computer vision), and his distance from the display (through an IR sensor in the screen bezel).

After brainstorming and sketching interaction options (Fig. 1A), the designer begins by extending an existing visualiza-

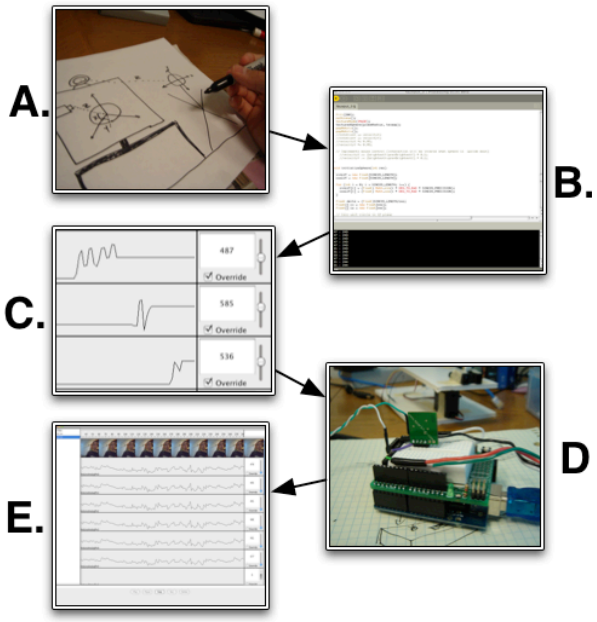

Figure 1. (A) Brainstorming and UI sketching; (B) Software prototyping; (C) Simulating hardware data in *FauxPut*; (D) Hardware prototyping; (E) Replaying inputs for testing.

tion code example (Fig. 1B). While our designer has access to a webcam for vision sensing, he has not yet built the IR distance-sensing apparatus. To test the visualization at this stage, he records a sample video with a webcam and then simulates IR sensor values in *FauxPut* through a GUI slider while replaying the video (Fig. 1C). Once he decides on the program's general behavior in response to this sensor data, our designer builds a basic hardware prototype (Fig. 1D) containing the IR sensor. He then records a new set of examples, now containing data from both the webcam and the IR sensor. Next he repeatedly feeds the recorded input traces to his software, making small changes with each iteration (Fig. 1E). In this way, he is able to perform tests for debugging purposes, as well as make other implementa-

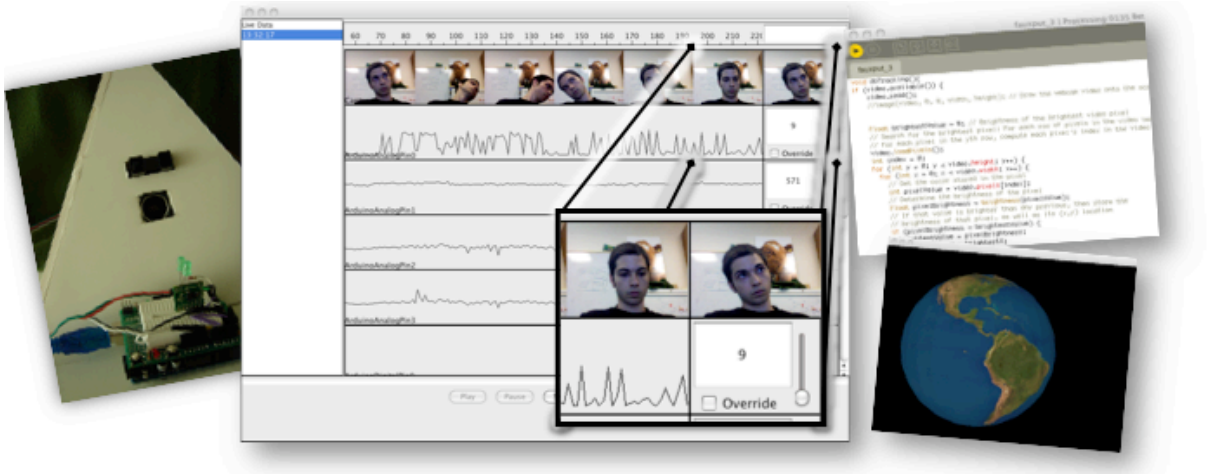

Figure 2. Left: a physical prototype with an IR sensor; Center: sensor data entering *FauxPut*; Right: the Processing IDE and a sample program receiving data from *FauxPut*.

tion changes, without having to physically perform the same input multiple times.

IMPLEMENTATION

We implemented *FauxPut* as an extension for Processing, a Java-based multimedia authoring environment popular with interaction designers and installation artists [2]. We currently support two types of input: camera streams, and sensor input from the Arduino platform [1]. Arduino was chosen for its flexible digital and analog sensing capabilities, while the camera device was selected as a powerful sensor with appealing library support within Processing.

In *FauxPut*, both the Arduino and video capture device data are presented to the user as individual timeline tracks. Each timeline may contain a graph of input data and a slider widget allowing the user to override the data received from the device (for Arduino devices) or a selection of frames (for video capture). The input data is displayed in real-time, giving the user visual feedback for both real and simulated inputs. A list of recorded traces on the left and a button panel on the bottom of the screen allows the user to record and play back the data received from these devices.

When the user's application attempts to read from an external device, the *FauxPut* wrapper classes return either live data, prerecorded data, or simulated data, allowing users to leverage *FauxPut*'s functionality without altering their code.

RELATED WORK

Prior research has investigated recording and simulation of input in isolation; to our knowledge, FauxPut is the first system that proposes to unify these functions in a visual direct manipulation interface aimed at testing.

Time-shifting: Substituting prerecorded sensor data for live input has been explored in DART [8], a Director-based authoring tool. Robots – software tools that emulate human input (e.g., java.awt.robot) – provide record and replay capabilities for mouse and keyboard.

Simulation & Proxy controls: Juxtapose [6] allows for real-time experimentation with program variables during execution. These variables can be derived from sensor data, but there is no way to record and replay real input data. In Phidgets [4], physical widgets could be used to control GUI widgets. *FauxPut* conversely uses GUI widgets to simulate the data that would be provided by physical inputs.

Visualization: Exemplar [5] uses signal visualization to enable users to select regions of input event streams. These regions are then used to train recognition algorithms and set event thresholds. Visual data flow environments such as Pd [3] or MaggLite [7] offer a way of inserting visualization or control widgets into the path of a data processing pipeline. However, these applications provide no means of controlling the data before it enters the pipeline.

ONGOING & FUTURE WORK

To complete the vision of a flexible input testing toolkit we are working to enable editing of previously recorded data within *FauxPut*. Presently, traces can only be overwritten in real time. Extending beyond the current library support for camera and sensor input, we plan to introduce an extension protocol that allows knowledgeable developers to add additional input devices, e.g., GPS or still camera control, to *FauxPut*.

REFERENCES

[1] "Arduino - HomePage"; http://www.arduino.cc/.

[2] "Processing 1.0 (BETA)"; http://processing.org/.

[3] "Pure Data — PD Community Site"; http://puredata.info/.

[4] S. Greenberg and C. Fitchett, "Phidgets: easy development of physical interfaces through physical widgets," *In UIST: ACM symposium on User interface software and technology*, 2001, pp. 209-218.

[5] B. Hartmann et al., "Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition," *In CHI: ACM conference on Human factors in computing systems*, 2007, pp. 145-154.

[6] B. Hartmann et al., "Design As Exploration: Creating Interface Alternatives through Parallel Authoring and Runtime Tuning," *In UIST: ACM Symposium on User Interface Software and Technology*, 2008.

[7] S. Huot et al., "The MaggLite post-WIMP toolkit: draw it, connect it and run it," *In UIST: ACM symposium on User interface software and technology*, 2004, pp. 257-266.

[8] B. MacIntyre et al., "DART: a toolkit for rapid design exploration of augmented reality experiences," *In UIST: ACM symposium on User interface software and technology*, ACM, 2004, pp. 197-206.