PocketTable: Mobile Devices as Multi-Touch Controllers for Tabletop Application Development

Stefan Hafeneger stefan.hafeneger@rwth-aachen.de
Malte Weiss weiss@cs.rwth-aachen.de
Gero Herkenrath gero@cs.rwth-aachen.de
Jan Borchers borchers@cs.rwth-aachen.de
RWTH Aachen University, Germany

Abstract

We present PocketTable, a generic framework for sending multi-touch events from external controllers to tabletop applications. Our system supports the development of multi-touch applications in a mobile context if no table hardware is available. Since PocketTable represents an abstraction layer between the device and the software, external devices can be interchanged without recompiling the code of the application.

1. Introduction

Multi-touch interaction tabletops have become a very active research area and many applications have been developed for them. However, writing tabletop applications still requires the programmer to work directly at the table. In the traditional setting, a computer renders the user interface, which is output by a projector, whereas a camera in the table provides multi-touch events, which are processed by the computer. Testing tabletop applications outside this *local context* was inconvenient to date.

PocketTable faces this issue by providing a framework which abstracts multi-touch events from different external devices and sends device-independent events to the application. Unlike existing approaches, PocketTable especially supports the *mobile context* since the small form factors of mobile devices are taken into consideration.

For example, using our system the developer can run her application on a laptop and use a multi-touch mobile device like the Apple iPhone for retrieving multi-touch events. After establishing a connection between the device and the laptop, the user determines the target area of the multi-touch events. Then all multi-touch events that are performed on the mobile device are directly sent to the remote application. The mapping can be changed at any time.

Due to the abstraction of events, the multi-touch application does not have to consider where the multi-touch events come from, simplifying switching between local and mobile context.

2. Related Work

[2] introduced the TUIO protocol, a network protocol for sending input events from various multi-touch interfaces to applications. This protocol provides independence from platform and programming language. However, it is unidirectional and therefore not suitable for the mobile context because it does not communicate the target area of the touch events.

Not many papers have been published on testing and debugging multi-touch applications outside the table environment. Although multi-touch simulators exist, as the TUIO simulator [1], they require the developer to use the mouse and do not provide real multi-touch input.

3. System Design

As shown in figure 1, PocketTable consists of several units that are responsible for communicating with the external devices and translating their incoming data into unified multi-touch events. These events are then dispatched into the native event handling system.

This abstraction from the low-level multi-touch implementation provides an application programming interface to the developer that does not differ for each input device. Thus, she does not have to care about the actual input device that will generate multi-touch events for the application.

With such a modular system we can use any mobile device with multi-touch capabilities during the development process without having to sit beside a tabletop system. This approach allows the programmer to develop multi-touch applications at any location.

On a multi-touch capable mobile device we need a client application that is able to connect to a PocketTable server. This client application then sends all multi-touch events to the server via a wired or wireless connection. On the server

the data is interpreted and touch events are dispatched into to the operating system.

3.1. Screen Size Mapping

There is an important downside with multi-touch capable mobile devices: The size of the screen is very small compared to multi-touch tables or screens in general. We face this problem with a bidirectional client-server connection. After it is established, the server sends its screen size and optionally a current screenshot to the mobile device. The client application then allows the user to define a ”target area” on the server's screen for the touch events to occur. Optical feedback to help with this calibration will be provided on the client device if possible. The client application is responsible for converting the local touch events according to the ”target area”. With this approach the server receives unified touch events and does not need to know any client device specific data. Since we need a bidirectional communication the TUIO protocol is inappropriate, since it does not consider arbitrary data. Extending the TUIO protocol to support this kind of additional data could be an option for the future.

4. Prototype

To demonstrate our approach, we built a multi-touch framework for Mac OS X that uses the native event handling system and the responder chain to dispatch multi-touch events to the applications. We wrote a client application for the iPhone to use it as mobile controller device. The connection is established via WiFi using Apple's zero configuration service discovery protocol Bonjour. Thus the programmer can create an ad-hoc network everywhere without having to deal with the task of establishing a client-server connection. Having examined the multi-touch architecture of Apple's multi-touch UI toolkit for the iPhone and iPod touch, we decided to create a similar architecture for our multi-touch extension. This way, the developer is working with a familiar application programming interface and a transition to a potential future native multi-touch implementation from Apple is likely quite easy.

5. Future Work

Due to our modular system approach we can now implement an input unit for a FTIR / DI multi-touch table without having to change the high level application programming interface. A general purpose unit supporting TUIO compatible devices would allow use of multi-touch devices that do not run a specific client application themselves. Multi-pen tablets could get their own unit as well.

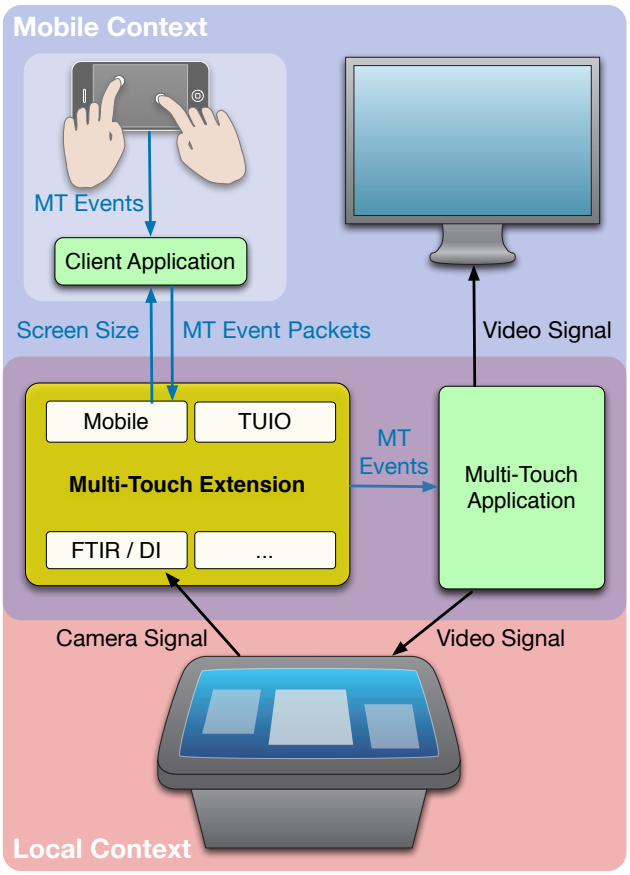

Figure 1. System Design Overview

Besides developing new units, we will extend our approach with a live-stream capability, where appropriate. This would allow the user to see the table activities directly on the mobile device, making the interaction more direct. Furthermore, we will conduct user studies for testing the usability and efficiency of controlling tabletops using mobile devices.

References

[1] M. Kaltenbrunner and R. Bencina. reacTIVision: a computer-vision framework for table-based tangible interaction. *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 69–74, New York, NY, USA, 2007. ACM.

[2] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO - A Protocol for Table Based Tangible User Interfaces. *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, Vannes, France, 2005.