A Usable API for Multi-Surface Systems

Chris Burns
Dept. of Computer Science
University of Calgary
Calgary, AB Canada
chris.burns@ucalgary.ca

Teddy Seyed
Dept. of Computer Science
University of Calgary
Calgary, AB Canada
teddy.seyed@ucalgary.ca

Theodore D. Hellmann
Dept. of Computer Science
University of Calgary
Calgary, AB Canada
tdhellma@ucalgary.ca

Mario Costa Sousa
Dept. of Computer Science
University of Calgary
Calgary, AB Canada
smcosta@ucalgary.ca

Frank Maurer
Dept. of Computer Science
University of Calgary
Calgary, AB Canada
frank.maurer@ucalgary.ca

Abstract

A multi-surface system brings together into a single application a wide variety of different devices. Interactions for these systems must be designed to span across all these devices – which could include tabletops, large format displays and mobile phones. This integration allows users to take advantage of the unique capabilities of each device in ways that would not be possible using those devices separately. However, creating usable interactions for moving content and control between such devices has proven a difficult problem. Gestural interactions, especially those informed by the spatial layout of the room as well as the people and devices in it, might provide a solution to this problem. But such systems are difficult and tedious to build and represent too large an investment of time and effort for developers to bear. To decrease the cost of developing such systems, we have created an API – called MSE-API – that allows developers to quickly and efficiently add gestural interactions to multi-surface applications. In this work we present the API, its uses cases and its structure.

Author Keywords

Multi-surface system, API Design, API Usability, Framework

ACM Classification Keywords
Design, Human Factors

Introduction

A multi-surface system allows developers to take advantage of the unique interaction capabilities of many different devices while still operating within a single distributed application. For example, a large digital tabletop provides a public work area in which users can work together to solve a problem while a tablet provides a workspace in which a single user can work privately from the rest of the group. In a multi-surface system, both of these interaction modes can be made available within the context of a single application. But in order for such a system to be usable, it must be straightforward for users to move content and control between various devices.

We propose that this problem can be solved by a multi-surface system where information transfer and control is informed by the spatial layout of the system itself. That is, by using the location and orientation of the devices in the environment where the system is in operation. Suppose a user wishes to transfer an image within a multi-surface system which contains a tabletop, a large format display and several others tablets held by other users. In our proposed system, a user would simply point their device towards their intended recipient and perform a *flick* gesture. The system will transfer the image to the device or devices in the user's field of view.

To support cross device interactions a spatial information a tracking system is necessary. Likewise, to support inter-device communication between devices robustly, a networking or message-passing framework must also be used. This means that the development of multi-surface systems requires specialized knowledge in a variety of fields. It would be possible to reduce this burden, however, with an API which provided each of these services for developers.

Building on our previous work in defining gestural interactions for multi-surface systems, we present an API for building multi-surface systems [1]. This API – MSE-API – handles several use cases critical to building multi-surface systems, including position and orientation tracking of people and devices in the room, data fusion, device discovery, and inter-device communication.

This paper presents MSE-API, gives a description its supported use cases and outlines its structure. We also propose future work for further development and evaluation-

Related Work

A core technical problem when building a multi-surface system is how to divide control and move content between component devices. Creating usable interactions for these tasks has been a major goal in multi-surface and multi-display research area for over two decades.

A novel solution to this problem involves the use of gestural interactions. To date, research in this area can be divided into: gestures performed with devices; gestures performed on devices; and gestures performed in physical space. The gestures proposed in this research tend to mimic physical interactions found in the real world, such as flicking and throwing to transfer content or pointing to make a selection.

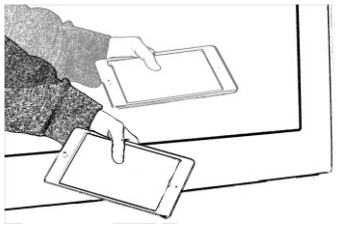

Figure 1. Gesture "with" a device.

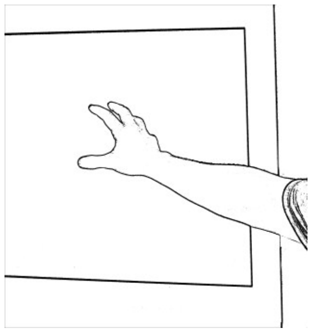

Figure 2. Gesture "without" a device.

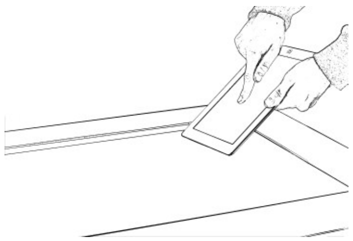

Figure 3. Gesture "on" a device.

Gestural Interactions
By tracking the movement of a device we can implement gestures which are performed *with* a device. Some of these gestures, for example a *throw* (See Figure 1), has been proposed to trigger content transfer [2, 3]. In our system these can be used as triggering actions but are augmented with a complete view of the spatial layout of the room so they can be used to both select the target device and initiate transferring of content.

Some gestures can by performed by users moving their hands and arms and these can also be used in a multi-surface system (See Figure 2). Work by Voida et al. proposed a *grab* and *point* gesture for selecting objects in an augmented reality environment [4]. By tracking the movement of a device rather than body positions we can implement gestures which are performed *with* a device. CodeSpace, an application created by Microsoft Research, incorporates several of these gestures to support information sharing during developer meetings [5]. For example, a *flick* gesture is used to transfer task information from a tablet device to a wall display (See Figure 3).

Previous APIs
APIs have been proposed in the literature for supporting specific aspects of building a multi-surface system.

Proximity toolkit provides developers with proxemic information, defined by the authors as position, identity, movement and orientation of people within the interaction space [6]. However, the framework is intended for the larger scope of *proxemic interactions* and is not focused on devices or multi-surface systems specifically. As the goal of MSE-API is to support real world applications, some of the constraints of Proximity Toolkit become major difficulties. Proximity toolkit works mainly with the highly expensive Vicon[1] tracking systems, which cost upwards of one-hundred thousand dollars. To be useful in real world situations MSE-API supports the Kinect which is available at consumer level prices. Both the Vicon system and the OptiTrack[2] system require markers to be attached to users and devices. Such markers would be inconvenient in real work applications.

The 3MF framework exposes features typically found in many devices in a multi-surface system (e.g. gyroscopes or accelerometers) through a common interface [7]. It also allows logical commands to be passed to different devices within the system. This framework has a relatively low level focus and does not concern itself with proximity or spatially augmented multi-surface system. It therefore doesn't provide spatial or location information.

Use Cases

MSE-API provides two distinct solutions for two types of use cases within a multi-surface system: *spatial query* tasks and *device communication tasks*. These allow developers to build multi-surface systems where gestural interactions are used to move content and control around the different devices in the system.

Spatial Queries
In MSE-API, position data (collected using a Microsoft Kinect) and orientation data (collected from individual

[1] http://www.vicon.com/products/

[2] http://www.naturalpoint.com/optitrack/

mobile devices) is centralized into a single server called the Locator. We provide convenience methods that allow developers to query the Locator based on two spatial properties: proximity and field of view.

Proximity queries allow developers to determine which devices are located within a given distance of the calling device. This functionality can be used in a variety of ways, including dispatching content to the mobile devices of all users working within a given area. Alternatively, it could be used to determine if a user was physically overlapping with another device – for example, if a user was holding their device over a digital tabletop. This functionality could be used to support a *flick* and a *pour* gesture respectively.

MSE-API also supports queries based on the orientation or field of view of a device. These queries can be used to detect which devices in the system fall within the field of view (as defined by the developer) of a given device. Using the field of view of a device is useful for situations where a user wishes to make a selection by pointing or selecting with their device. For example, consider a situation in which a user wishes to send a picture to another device in a system. The user would select the receiver of this content by orienting their device towards their target; effectively allowing the API to determine, out of many possibilities, which single target device should receive the picture.

These queries represent common functionality that is needed to implement many of the gestures that could be useful in a multi-surface system [1]. We expect that developers will use these queries to compose a wider variety of gestures then we have yet to propose.

Device Communication

Once a gesture has been detected and the developer has queried the Locator for the intended device, messages must still be forwarded to the target device to complete the interaction. In MSE-API, the message passing framework is closely integrated with the rest of the API. As this communication is achieved using standard web techniques (HTTP messages and RESTful APIs) it is possible for developers to quickly integrate powerful message passing functionality directly into the application.

Structure of API

MSE-API can be divided into two major components: the Locator and the Client Libraries. The Locator provides spatial information about devices and people in the room. The Client Libraries provide necessary information to the Locator and also make communication between the devices more convenient and straightforward. This is consistent with MSE-API's main design goal of "making easy things easy, and complex things possible."

Locator

The Locator maintains a complete model of the positions of every device in the room, including position-fixed devices (e.g. a tabletop or wall display) and the positions of mobile devices (e.g. tablets and mobile phones). While fixed devices can be positioned manually when setting up the system, it is necessary to track mobile devices as they move about the room. Rather than track the devices themselves, MSE-API tracks the positions of people in the room and *pairs* an individual device with the person holding that device.

This pairing relationship is initialized when a user performs a waving gesture where both the waving motion of the device and the waving motion of the person's hand are detected. The position of the person, which is easy to track using consumer hardware such as the Kinect, can approximate the position of the device. This relationship, however, is broken when a user is occluded or leaves the tracked area of the room.

In addition to position, the Locator is also updated with the orientation of the devices in the system. This information is captured by using the gyroscope provided by most mobile devices. For the purposes of selection and targeting actions, this orientation is used as proxy value for the orientation of the person.

The API currently uses a Kinect to provide tracking information about users in the room, but, since tracking information is exposed via a REST service which can be updated using HTTP requests, alternative hardware such as the OptiTrack system[3] or the Vicon tracking system[4] could be easily substituted.

Client Components

The Client Libraries are components of the API that developers can integrate into applications running on the various devices in a multi-surface system (e.g. a C# application running on a Microsoft Surface or an Objective-C application running on an iPad). These provide two major pieces of functionality: integration with the Locator; and integration with other devices.

[3] http://www.naturalpoint.com/optitrack/

[4] http://www.vicon.com/

The Client Libraries provide all the required information to the locator without development effort for those using the API. This includes updating the pairing state of the device and detecting the *wave* gesture which a user performs to start using the system. The Client Library also captures orientation information from the device and provides this information to the Locator. Lastly, it provides a set of convenience methods for querying the Locator without writing networking code.

To support communication between devices in the room, MSE-API provides several convenience methods for sending data to other devices. This includes common data types such as binary data, images, and dictionaries. Likewise, developers can specify callback methods for when they receive one of these data types. For more advanced use cases, MSE-API provides the ability to create HTTP routes and process the results of HTTP requests. This functionality is implemented using a networking and device discovery framework called IntAirAct.[5]

Future Work

We plan to expand our research in two specific directions: the further evaluation of MSE-API and its further technical development. We are currently conducting a longitudinal case study in which MSE-API is being used as part of a different multi-surface system currently under development.

We would also like to add additional interactions to those currently supported by MSE-API and increase the range of its tracked area by adding support for multiple Kinect sensors.

[5] http://arlol.github.com/intairact.html

Conclusion

Finding usable interactions for moving content and control through a multi-surface system has been a long-standing goal of multi-surface research. New gestural interactions, which are based on physical actions in the real world, such as a *throw* gestures to transfer content or a *point* gesture to support selection have been proposed by researchers. In this paper we have presented MSE-API, which provides these interactions for multi-surface systems. We describe the major use cases and structure of the API and propose future work for its evaluation and further development.

References

[1] A. Bragdon, R. DeLine, K. Hinckley and R. M. Morris , "Code space: touch + air gesture hybrid interactions for supporting developer meetings," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS'11)*, Kobe, Japan, 2011.

[2] R. Dachselt and R. Buchholz, "Natural throw and tilt interaction between mobile phones and distant displays," in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, Boston, MA, USA, 2009.

[3] T. Döring, A. S. Shirazi and A. Schmidt, "Exploring gesture-based interaction techniques in multi-display environments with mobile phones and a multi-touch table," in *Proceedings of the International Conference on Advanced Visual Interfaces (AVI'10)*, Rome, Italy, 2010.

[4] B. Kaufmann, M. Gratzer and M. Hitz, "3MF – A Service-Oriented Mobile Multimodal Interaction Framework," in *Proceedings of the Workshop on infrastructure and design challenges of coupled display visual interfaces (PPD'12)*, Capri, Italy, 2012.

[5] A. Bragdon, R. DeLine, K. Hinckley and R. M. Morris , "Code space: touch + air gesture hybrid interactions for supporting developer meetings," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS'11)*, Kobe, Japan, 2011.

[6] N. Marquardt, R. Diaz-Marino, S. Boring and S. Greenberg, "The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies," in *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST'11)*, Santa Barbara, California, USA, 2011.

[7] T. Seyed, C. Burns, M. C. Sousa, F. Maurer and A. Tang , "Eliciting usable gestures for multi-display environments," in *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces (ITS'12)*, Cambridge, Massachusetts, USA, 2012.