HAL
open science

ModControl – Mobile Phones as a Versatile Interaction Device for Large Screen Applications

Matthias Deller, Achim Ebert

To cite this version:

Matthias Deller, Achim Ebert. ModControl – Mobile Phones as a Versatile Interaction Device for Large Screen Applications. 13th International Conference on Human-Computer Interaction (INTERACT), Sep 2011, Lisbon, Portugal. pp.289-296, 10.1007/978-3-642-23771-3_22 . hal-01590869

HAL Id: hal-01590869

https://inria.hal.science/hal-01590869

Submitted on 20 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

ModControl – Mobile Phones as a Versatile Interaction Device for Large Screen Applications

Matthias Deller[1,2], Achim Ebert[1]

[1] University of Kaiserslautern, Gottlieb-Daimler-Straße 47,
67663 Kaiserslautern, Germany
[2] DFKI GmbH, Trippstadter Straße 122,
67663 Kaiserslautern, Germany
Matthias.Deller@dfki.de, Ebert@cs.uni-kl.de

Abstract. Large, public displays are increasingly popular in today's society. For the most part, however, these displays are purely used for information or multimedia presentation, without the possibility of interaction for viewers. On the other hand, personal mobile devices are becoming more and more ubiquitous. Though there are efforts to combine large screens with mobile devices, the approaches are mostly focused on mobiles as control devices, or they are fitted to specific applications. In this paper, we present the ModControl framework, a configurable, modular communication structure that enables large screen applications to connect with personal mobile devices and request a set of configurable modules, utilizing the device as a personalized mobile interface. The main application can easily make use of the highly sophisticated interaction features provided by modern mobile phones. This facilitates new, interactive appealing visualizations that can be actively controlled with an intuitive, unified interface by single or multiple users.

Keywords: Interaction framework, Distributed interfaces, Input devices and strategies, User-Centered Design.

1 Introduction

Today, there are two fast growing trends in display types, away from the conventional desktop screen paradigm.

On the one hand, there is an ever increasing number and variety of small screens, on personal mobile devices like smart phones, netbooks or, lately, tablet computers. Screens on mobile devices keep getting more sophisticated, both in terms of resolution and in interaction and navigation possibilities. Multitouch interaction is rapidly becoming the de facto standard for smart phones, enabling users to control the device with intuitive finger gestures and reducing the need for hardware keys by replacing them with configurable soft keyboards. In addition, more and more portable devices come equipped with auxiliary input methods, like accelerometers and/or magnetometers, facilitating varied new forms of interaction. These devices are

ubiquitous, unobtrusive, versatile, and personalized to the information and preferences of their owners.

The drawbacks of these devices are the limits imposed by hardware and physical restrictions. Although mobile processors are becoming faster, the need for saving energy puts restraints on the complexity of computing tasks achievable with the device. Also, the memory available on smart phones is extremely limited, impeding works with large data sets. Finally, the size of the display puts natural hindrances on complex visualizations.

On the other hand, physically large displays are becoming more and more common, to the point of being almost omnipresent. Their forms range from large personal screens like home theatre screens to public screens for advertising or education. A problem with such screens is that the usual interaction metaphors with keyboard and mouse are no longer adequate for interacting. For one, large displays usually include an area in front of the display, in which viewers can move to get an overview or to obtain a closer look on details. Consequentially, interaction devices for large screens should support this kind of mobility, prohibiting stationary devices like keyboards or mice. In the case of public screens, there is often the problem that the screens are not physically reachable by users, either because they are out of the user's range, or because accessibility is restricted to prevent vandalism. In these cases, touch interaction is also not an option to interact with the application.

While there are approaches to combine large displays and mobile devices, they are mainly focused on either using the mobile as a pure controlling device for the display, or using the large screen as auxiliary display for the mobile device. In this paper, we present ModControl, a fusion of both large display and mobile device interaction. With it, we provide a flexible and versatile interface for mobile devices to interact with large screen applications. Using an XML communication framework and module-based, configurable clients for mobile devices, we enable sophisticated and intuitive metaphors for large screen interaction, while providing users with personalized access to collaborative large screens. Possible application scenarios include electronic pin boards, private information retrieval for public displays, or appealing entertainment applications like collaborative games.

2 Related Work

The idea of combining large public displays with personal mobile devices has been around for some years now. In most cases, research can be divided in two categories, determined by the direction of the flow of information. In one case, the main part of the application is carried out on the mobile device, with the large screen as auxiliary output devices. In the other case, the mobile device is used as control device for a large screen application or other systems that can't be interacted with directly.

The WebSplitter project [5] offers interaction in the output direction, by specifying different end-point devices as targets for web-based multimedia content. Based on rules created via an XML protocol, specific multimedia content can be sent to corresponding devices. Connected devices are in this case used only as output for the content, without being able to provide feedback to the system.

Conversely, the “inverted browser” of Raghunath et al. [9] is a browser application running on the mobile device. A network service running on the computer driving the large screen enables the mobile device to push content from the browser application to be displayed on the large screen. This approach is motivated by the idea of symbiotic display environments [3].

A different approach for multiple display environments is used by the Universal Interaction Controller of Slay and Thomas [11]. The UIC consists of a spatially aware mobile device, the Ukey, an interaction manager, and a clipboard manager. With the Ukey, the user can select an active screen from the connected displays, and then interact with the contents of the display using the handheld device. Movement of objects on one display is possible, but the main benefit of the system is to be able to transfer objects between screens by copy and paste.

Another type of connection between large screens and mobile devices is a method called “peephole navigation” [12]. This technique requires a spatially aware mobile device. By moving it around in two or three dimensions, the user can utilize it as a metaphorical “window” to different parts of a large virtual workspace. At the same time, the pen interaction of the PDA can be used to manipulate said environment.

Myers et al. developed the Pebbles system, which consisted of several Palm Pilot PDAs attached to a single PC via serial connections, enabling several users to interact with the system simultaneously. However, interaction was restricted to low-level events sent from the PDAs to the PC. Later, the authors extended their approach with the “Semantic Snarfing” technique [7]. Here, users could interact remotely, using a combination of PDAs and Laser Pointers. In this manner, it was possible to select images, menus or text items on the PC screen, causing an adapted copy to appear on the PDA. Here, they could be interacted with appropriately and subsequently sent back to the main application.

Another possibility adopted by several projects is to use the camera integrated in most mobile phones for interaction. Madhavapeddy et al. [6] use spot codes to tag objects displayed on the large screen. By analyzing the image of the phone’s camera, they can identify which tag the device is currently pointed at, as well as additional information encoded in the tags. Using this information, the user can navigate the application by selecting the appropriate objects/tags.

A similar method is presented by Ballagas et al. in [2]. The point & shoot interaction technique is also based on visual codes embedded in the main visualization. In this case, however, the codes are not attached to objects, but rather form a grid covering the whole display area. Since the grid of visual codes is occluding a lot of information contained in the original visualization, the codes are only displayed for a short time span following the user triggering a selection. Another technique the authors describe is the sweep method. Here, the optical flow of the camera’s image resulting from movement of the device is analyzed to implement a relative movement control for the mouse cursor.

A comprehensive list of possible interaction subtasks using mobile devices as well as different approaches to address these tasks can be found in [1].

More recent research, e.g. [10] and [4], investigates the performance of interfaces using the same approach as ModControl, namely distributing the interface between a large screen application and one or more small screens that are not only used to control the main application, but also to complement the large display with a small,

private screen. The authors of [4] conducted a user study with three different applications to compare performance of interaction widgets that are distributed on large and small screens with widgets that are placed solely on a large display. While overall performance was equal in both cases, the degree of user satisfaction was higher and the error rates lower in the large screen/small screen scenario.

3 The ModControl Framework

Principally, the ModControl Framework is a Server-Client setup. A message server is responsible to keep track of the connected clients and passing messages between them. Received messages are put on a FIFO stack and then distributed to the corresponding clients, which can be categorized in application clients and interaction clients.

Application clients are clients that require input or data from other clients, like a large public screen without inherent interaction possibilities. In the majority of cases, there would be only one application client connected to the server, but the framework supports different application clients making use of the same network.

The interaction clients, on the other hand, are modularized apps running on the user’s personal mobile device. They enable users to interact with the main application by providing information or input data from the mobile device or present specific and personalized data on it. For this, the main application can send requests for specific modules to the interaction clients and register for events sent by these modules.

3.1 Communication and Message Format

The communication messages of ModControl are in XML format, making it easy to add additional, application-specific messages.

All Messages have a root element of the type msg. This element has the required attributes class and type. Another attribute that can be set by the message server is the sender attribute, which contains the sending client's ID that has been assigned when the client first connected. Additionally, the target-attribute can be included to send messages to specific clients.

Basically, the class attribute can be used by an application to group messages into specific subgroups, or to distinguish messages of different main application using the same message server. The only exception is the sys class, which the ModControl framework reserves for network management like pings and registering clients for specific message types.

The order in which the clients connect to the server is irrelevant, but in most cases the application client will be the first client to connect and start listening for connecting interaction clients. In both cases, the connection is handled by the XMLClient. This class capsules all required low-level functionality.

The application client should derive a customized class to provide callback functions for received interaction messages. As soon as an interaction client connects, the application client can send it a message of the type needModule to indicate which types of interaction modules are supported by the application.

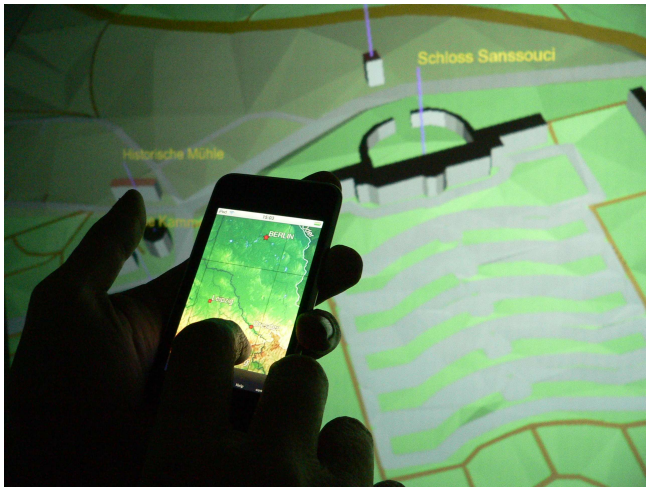

Figure 1. Users can control applications without obstructing the main visualization.

3.2 Modular Client and Implemented Modules

The interaction client running on the mobile device is essentially a wrapper taking care of server connection and communication, and housing the modules requested by the application client. This is done by creating tabs for each active module. Whenever a needModule message is received, a new tab for the module with the corresponding creation parameters is added to the client. This message has the required parameters hidden and priority, determining whether the new module should be visible in the tab list. The higher the module's priority, the higher up in the list it will appear. Also, the needModule message can define a module identifier, and a label that will represent it in the tab list. The main application can also switch to a specific tab by sending a showModule message to the interaction client. Changing the active tab on the client will result in a shownModuleChanged message that will be sent to all clients that have registered for that message type.

Connection Module: This is the initial module always available on the interaction client. It is used to establish the connection to an XML message server or to end an already existing connection. The tab for the connection module consists simply of two text input fields for server address and desired user name.

Text Module: As the name implies, this module is intended to either show fixed text information, or enable editing a text field. The text for the module can be set by including a text element at creation in the corresponding needModule message, or by a separate setText message. If the user has finished editing or changed the active module, a textChanged message containing the new text is sent to registered clients.

Mouse Emulation Module: With this module, a touch capable interaction client is used to emulate a touch pad. The display area of the tab is used to track the user's finger movements and translated into relative movement events for the main application's cursor. At the creation of the module, the application client can include a

numberOfButtons element in the needModule message to add one to three virtual mouse buttons to the touch pad area. A touchBegan message is generated whenever a new finger touches the pad area or the virtual buttons, containing the coordinates of the touch, a tap count, the timestamp and, if applicable, the index of the pressed button. If an existing touch is moved around the pad area, a touchMoved with similar parameters is generated. Finally, a touchEnded message is created for each finger leaving the pad area.

Image Selection Module: The image selection module can be used to display arbitrary images on the mobile device's display. When creating the module with a needModule message, the application client can include an imageData element containing the image encoded with base 64 RFC 3548. In case the image is too large to fit in the mobile's display area, the application client can also determine x and y offset to ensure that a specific part of the image is shown. Users can use the mobile's capabilities to rotate, pan or zoom. Whenever the position or scale factor of the image is changed, an according message with the updated value is sent. By using a double tap on the currently shown image part, the user can generate an imageTapped message. The message includes the current orientation, offset and scaling factor of the image on the mobile's screen. Additionally, it contains a timestamp, the touch's position on the screen, the number of taps, and finally, the coordinates of the touch in the image's coordinate system. It is also possible to use the mobile client to facilitate multi-user picking functionality for the main application by sending a screenshot to the mobile device and use the returned tapping coordinates for object picking.

SwitchList Module: Another important module for the mobile client is the switch list. It is used to control a collection of on/off type parameters. The application client has to provide a list with elements that constitute the switches displayed in the list. The state of one or more entries can also be set subsequently with a setStates message. Whenever one of the states is toggled, the client generates a switchChanged message containing the name of the switch and the new state.

Accelerometer / Magnetometer Module: The accelerometer/magnetometer modules are the only modules that have a permanent hidden status, meaning they will not appear in the module tab. The only parameter that can be given for these modules is the desired update rate per second. The data itself is sent by an accelerometerData or magnetometerData message, respectively. Both messages include a data element containing the x, y, and z values for the corresponding sensor as well as a timestamp.

3D Navigation Module: The most specific of the currently implemented modules is a module for navigating in 3D surroundings. It is intended to provide a fly- or walkthrough metaphor navigation for virtual environments utilizing both touch recognition and accelerometer data. The module's tab on the client consists of a navigation dial indicating the major movement directions. Users can move forward and backward by sliding their finger along the vertical axis, determining direction and speed of the movement. By moving the finger along the horizontal axis, a strafing movement is controlled in the same manner. By tapping and holding the buttons at the end of the horizontal axis, the virtual viewpoint can be rotated. To enable movement along the third axis, the user can activate the acceleration-controlled fly mode by tapping a button below the navigation dial. This causes the module to send additional accelerometer data that can be used to control the pitch and yaw orientation.

4 Demonstration Application

As an initial demonstration and evaluation scenario, we decided to use an existing real world application that has need of several visual interfaces even on a normal desktop computer. The application we decided on is a large screen visualization of the OpenStreetMap-3D dataset [8]. It offers a Web 3D service providing a Digital Elevation model of Germany as a 3D scene graph. Our initial application was an implementation to visualize said data interactively on a large stereoscopic screen. To navigate the 3D environment, the user could use a fly through metaphor. This was done with the standard combination of keyboard and mouse that is used in most 3D games. The speed of the movement was fixed, but could be adjusted using the keyboard. Another key would superimpose a map of Germany over the main visualization, highlighting the current position. Moving the mouse to select a target location and clicking on it, the user could quickly teleport to this location.

The overall scene graph of OSM-3D consists of several layers. In addition to the basic DEM data, there are layers for building models, labels for cities, and so on. These could be toggled to be shown in the visualization by pressing another function key on the keyboard. This would bring up a checklist where users could toggle visibility of the layers. Using hotkeys, the user could also display various status values of the application, such as a FPS counter, the currently loaded tiles etc.

This led to several problems with regard to interaction with the application. For one, navigating the environment was rather awkward, forcing users to stand at a fixed position with keyboard and mouse. Also, they had to memorize keyboard shortcuts to invoke functions like the layer list or the teleport map. Also, visualizations like the map or the checklists occluded most of the actual visualization.

To alleviate these problems, we applied the ModControl framework to create a new distributed interface and facilitate a more intuitive and flexible navigation for the large screen application. The user can now intuitively control the movement of the virtual camera using only one hand while moving around in front of the display. Also, the speed of the movement is controlled continuously without the need of adjusting it via additional buttons. The teleportation functionality is realized by using the image selection module of the framework. When the corresponding tab on the mobile device is activated, the application client will generate a texture of the map and send it to the interaction client. There, the user can navigate on the map and select the target of the teleportation without interfering with the main visualization. By additionally employing the accelerometer module, the user can present the map to others by making a “pushing” motion towards the large screen, causing the map to be shown over the main visualization. The switching of layers and information overlays was transferred to corresponding switchlist modules on the mobile client.

Furthermore, several users with mobile interaction clients can control the system at the same time. To avoid confusion and conflicts in navigation, the features controlling the virtual camera are restricted to the first connected interaction client, while other interactions like the toggling of layers can be controlled by all users.

5 Conclusion and Future Work

In this paper, we presented ModControl, a flexible and versatile framework that enables personalized multi-user interaction with large screens using a module-based client on a mobile device. Applications can request specific modules on the mobile and so take advantage of the sophisticated interaction possibilities of modern smart phones. By using a message server/client structure, an application can also address multiple clients at the same time, enabling personalized interaction for multiple users.

Currently, we are designing more specific evaluation scenarios to be used in a formal user study of our system. Also, we aim to provide additional interaction modules for other features of mobiles, e.g. supporting cameras or multitouch gestures.

References

1. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G.: The Smart Phone: A Ubiquitous Input Device. IEEE Pervasive Computing, Volume 5, Issue 1, IEEE Computer Society (2006), 70.
2. Ballagas, R., Rohs, M., Sheridan, J.G.: Sweep and point and shoot: phonecam-based interactions for large public displays. In CHI '05 extended abstracts on Human factors in computing systems, ACM Press (2005), 1200-1203.
3. Berger, S., Kjieldsen, R., Narayanaswami, C., Pinhanez, C., Podlaseck, M., Raghunath, M.: Using Symbiotic Displays to View Sensitive Information in Public. In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society (2005), 139-148.
4. Finke, M., Kaviani, N., Wang, I., Tsao,V., Fels, S., Lea, R.: Investigating distributed user interfaces across interactive large displays and mobile devices. In Proceedings of the International Conference on Advanced Visual Interfaces, ACM Press (2010), 413-413.
5. Han, R., Perret, V., Naghshineh, M.: WebSplitter: A unified XML framework for multi-device collaborative Web browsing. In CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM Press (2000), 221-230.
6. Madhavapeddy, A., Scott, D., Sharp, R., Upton, E.: Using Camera-Phones to Enhance Human-Computer Interaction. In Adjunct proceedings of The Sixth International Conference on Ubiquitous Computing, ACM Press (2004).
7. Myers, B.A., Peck, C.H., Nichols, J., Kong, D., Miller, R.: Interacting at a Distance Using Semantic Snarfing. In Proceedings of the 3rd International conference on Ubiquitous Computing, Springer Verlag (2001), 305-314.
8. OSM-3D Germany. http://www.osm-3d.org/home.en.htm
9. Raghunath, M., Ravi, N., Rosu, M.-C., Narayanaswami, C.: Inverted Browser: A Novel Approach towards Display Symbiosis. Proceedings of the 4th IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society (2006), 71-76.
10. Sas, C., Dix, A.: Designing and evaluating mobile phone-based interaction with public displays. In CHI '08 Extended Abstracts on Human Factors in Computing Systems, ACM Press, 3941-3944.
11. Slay, H., Thomas, B.H.: Evaluation of a universal interaction and control device for use within multiple heterogeneous display environments. In Proceedings of the 7th Australasian User interface conference - Volume 50, Australian Computer Society (2006), 129-136.
12. Yee, K.-P.: Peephole displays: pen interaction on spatially aware handheld computers. In Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press (2003), 1-8.