Your Browser is the Controller - Advanced Web-Based Smartphone Remote Controls for Public Screens

Matthias Baldauf[1], **Florence Adegeye**[2], **Florian Alt**[3], **Johannes Harms**[1]

[1]Vienna Univ. of Technology
INSO Research Group
Wiedner Hauptstrasse 76
1040 Wien, Austria
{first.second}@inso.tuwien.ac.at

[2]FTW Telecommunications
Research Center Vienna
Donau-City-Strasse 1
1220 Wien, Austria
adegeye@ftw.at

[3]LMU München
Media Informatics Group
Amalienstrasse 17
80333 München, Germany
florian.alt@ifi.lmu.de

ABSTRACT

In recent years, a lot of research focused on using smartphones as input devices for distant screens, in many cases by means of native applications. At the same time, prior work often ignored the downsides of native applications for practical usage, such as the need for download and the required installation process. This hampers the spontaneous use of an interactive service. To address the aforementioned drawbacks, we introduce *ATREUS*, an open-source framework which enables creating and provisioning manifold mobile remote controls as plain web applications. We describe the basic architecture of ATREUS and present four functional remote controls realized using the framework. Two sophisticated controls, the Mini Video and the Smart Lens approach, have been previously implemented as native applications only. Furthermore, we report on lessons learned for realizing web-based remote controls during functional tests and finally present the results of an informal user study.

Author Keywords

remote control, public display, smartphone, interaction

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Input devices and strategies, prototyping*

INTRODUCTION

Digital public displays have come a long way from a static advertising medium to ubiquitous, networked screens that enable engaging experiences through interactive content. Today, different interaction modalities are commonly used, including touch, mid-air gestures, and smartphone-based interaction [11]. In particular, we see potential in the latter modality for that it does not require additional technology to be deployed at the display (touch surface, depth sensor, etc.).

At the same time, smartphone-based interaction infers several challenges, both for the user as well as for the display provider. From a user perspective, current solutions usually require downloading and installing a native application that serves as a remote control. Solutions range from generic remote controls apps (e.g., *Unified Remote*[1], depicted in Figure 1, or *HippoRemote*[2]) to mobile applications that integrate with a display application [1, 2]. Such tedious download and installation processes can even be necessary for different applications and for different screens. For providers, the obvious drawback of native apps is the need to develop them for multiple mobile platforms and that the functionality needs to be re-implemented for each application.

Figure 1: *Unified Remote* as an example for a native remote control application.

As a solution to these challenges, we present *ATREUS*[3] (Advanced web Technologies for REmotely controlling Ubiquitous Screens), a web-based open source framework that simplifies the dynamic provisioning of novel remote controls for multi-application displays. The framework provides high-level methods to exchange remote control commands and feedback by an interactive display application. Utilizing the manifold features of HTML5, advanced platform-independent mobile controls can be realized.

Stakeholders can benefit from this approach in several ways. For display providers, the costs of development, customiza-

PerDis '16, June 20-22, 2016, Oulu, Finland
2016 ACM. ISBN 978-1-4503-4366-4/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2914920.2915026

[1]http://www.unifiedremote.com/
[2]http://hipporemote.com/
[3]https://bitbucket.org/matbal/atreus

tion, and maintenance of native applications for several platforms is reduced. Application developers can focus on the actual interactive service instead of caring about the communication with and management of mobile devices. Users can be provided easy means to interact with a display, for example by scanning a QR code that directs to a website with the remote control functionality. And finally, also researchers can benefit, since the framework enables rapid prototyping of mobile remote controls and display applications without the need for re-compiling in short iteration cycles.

The contribution of this work is twofold. First, we describe the framework architecture and the implementation of advanced demonstrators which partly have been realized as native applications only so far. Second, we present lessons learned for the development of web-based remote controls and report on user feedback gained in an informal pilot study.

RELATED WORK

Our work draws from several strands of prior research, most importantly mobile interaction with displays and toolkits used in the context of interactive screens.

Mobile Interaction with Public Displays

Smartphones and feature phones have been used in different ways for interaction with large displays [6, 10]. Basic forms include using the traditional communication functionality of the device. For example, SMS has been used for text-based control of a display [13].

More sophisticated applications use smartphones as remote controls. For example, Vajk et al. used the accelerometers of a smartphone in a native application to control a racing game on a remote screen [20]. In a similar vein, Pietroszek et al. exploit built-in orientation sensors to enable various operations in a 3d environment on a distant screen through a smartphone [16]. An example for direct interaction is *TouchProjector*, where remote displays could be controlled by using the mobile device as a "Smart Lens", i.e. remote content could be manipulated through the live video image on the mobile device [7]. Sahami et al. used the flashlight of a phone to create a novel pointing device [19].

She et al. suggested to use the smartphones' accelerometers to detect gestures and hence indicate the selection of items on a display [18]. The user experience and performance of different remote control approaches was investigated by Baldauf et al. [3]. They compared touchpad-like and pointing-based smartphone controls with advanced techniques such as the camera-based Smart Lens approach.

Purely browser-based approaches for interacting with screens through a mobile device are scarce and are either isolated proof-of-concept applications such as a maze game[4] or a sports game[5] by Google or are restricted to simple non-dynamic use cases such as submitting a vote [5]. Most related to the work presented in this paper is *uCanvas* [12], a recent Web-based framework for using smartphones as gestural controllers for applications on a distant screen. In contrast, ATREUS does not focus on a particular type of remote control but enables different further kinds of advanced remote controls.

Toolkits for Display Applications

A number of toolkits and frameworks has been proposed with the aim to facilitate developing applications for interactive public displays.

For example, Cardoso et al. presented PureWidgets, a web-based toolkit that supports the development of interactive multi-display applications [9]. Their focus is on facilitating the creation of the UI by means of web-based widgets rather than on interaction. Linden et al. developed a web-based framework with the aim to facilitate the dynamic partitioning of screen real estate into virtual screens to concurrently run applications on an interactive display [15]. Schneegass et al. presented SenScreen, a toolkit to support the use of display sensors by multiple applications [17]. Realizing interaction is left to the designer of each application.

A technical approach similar to ATREUS was presented by Cardoso and Barreira [8]. Their public display toolkit also enables web-based smartphone controls, however, is currently restricted to basic remote controls such as a preliminary joystick control or text input. Finally, the Tandem Browsing Toolkit [14] allows building web applications with additional "virtual screens" to be displayed concurrently on mobile devices. The toolkit focuses on collaborative browsing and orchestrating navigation through online content. In contrast, ATREUS emphasizes the creation of advanced mobile remote controls which support streaming of commands for highly dynamic interactive applications.

While prior research focused on creating the UI and on providing raw sensor data, we present a toolkit that focuses on smartphones as universal input devices and enables realizing advanced remote controls in a platform-independent manner.

DESIGN REQUIREMENTS

At the outset of our research, we identified design requirement for the framework. The requirements are based on a literature review in Google Scholar with the keywords "smartphone mobile handheld interaction public screen" and an analysis of available native apps in Google's *Play Store* with the keywords "remote control gamepad". Additionally, we drew requirements from own experiences in creating remote control prototypes for mobile devices [1, 2, 3, 5].

R1. Web-based user interfaces. This is the most central requirement of the presented framework. To realize the aforementioned benefits for developers and display operators, the ATREUS framework should allow for the development of remote controls as web pages, i.e. using common web technologies such as HTML, CSS, and JavaScript.

R2. Stream-based bidirectional communication. Instead of single commands, the framework should allow for quick sequences of remote control commands to support highly dynamic applications such as games and enable feedback by the

[4] https://chrome.com/maze/

[5] https://chrome.com/supersyncsports/

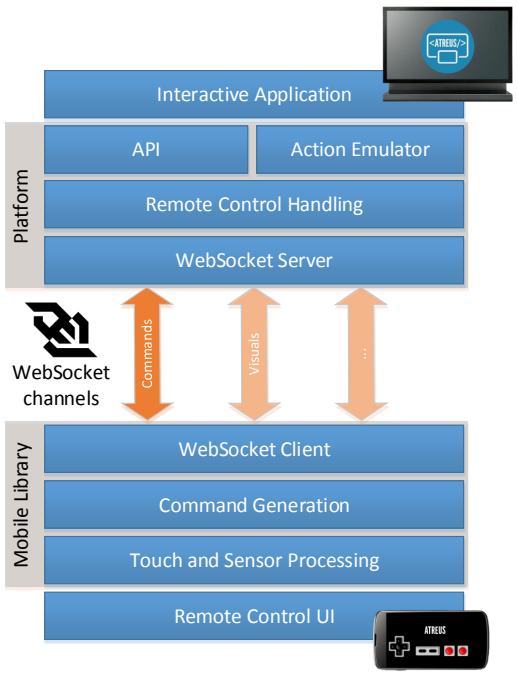


Figure 2: ATREUS architecture overview: The *ATREUS Platform* handles incoming commands and provides them to the interactive application running on the display. The *Mobile Library* enables the communication with the platform and contains methods to create control messages.

application. Such feedback could include vibration in case a player's character was hit or coupons in case a new high score was reached.

R3. Formalization of commands. Due to the lack of a suitable open protocol, we aim for a simple and generic way to formalize and exchange remote control commands between mobile devices and a remote application.

R4. Direct integration through API. Developers of applications for a public display should be able to add interactive features by connecting to the ATREUS framework through an easy-to-use Application Programming Interface (API).

R5. Support of legacy applications. For supporting existing applications and extending them with mobile remote controls, the framework should provide means to externally control such legacy applications, for example, by emulating respective key presses or mouse actions.

R6. Multi-user support. Our framework must provide methods to separate and react on commands issued from different mobile devices. Enabling interactive applications with several concurrent users is obviously one of the key advantages of handheld-based screen interactions over alternatives such as touch- or gesture-based approaches.

BASIC ARCHITECTURE

Similar to prior native approaches, we follow a typical client-server model (Figure 2): the server component, called *ATREUS Platform*, is executed on the computer hosting the large screen and features a QR code generator to enable the pairing process. Building mobile clients is facilitated through the *ATREUS Mobile Library*. These two components exchange commands via a WebSocket channel, a TCP-based full-duplex communication protocol implemented by modern web browsers. Information about commands is structured in the text-based JSON (JavaScript Object Notation) format, a de-facto standard for data-interchange in web applications. Advanced remote controls can open additional WebSocket channels for dedicated tasks (see examples in next section).

Platform

The ATREUS platform is implemented in *Java*. A *Jetty*[6] server receives commands from mobile clients. *Jetty* provides an open source HTTP web server and supports various extensions such as WebSockets. We encapsulated all central server-side functions in form of the *AtreusServer* class which can be easily integrated into applications to allow access from mobile clients and thus to enable interactive smartphone features. We applied the common observer pattern to specify operations to be triggered in case of relevant events such as incoming commands or a recently joining or leaving mobile client. Typically, commands are mapped to suitable key presses as specified in a configuration file by an *Action Emulator*.

Each exchanged command contains an interaction type (for example, button, touch, orientation, acceleration), an action (for example, up, down, move), the id of the associated UI element, a coordinate triple (typically for the touch point), and a multi-purpose value array (for example, for sensor data). Furthermore, each of these objects features a combination of IP and port of the source device as client identifier to distinguish different devices for multi-user applications.

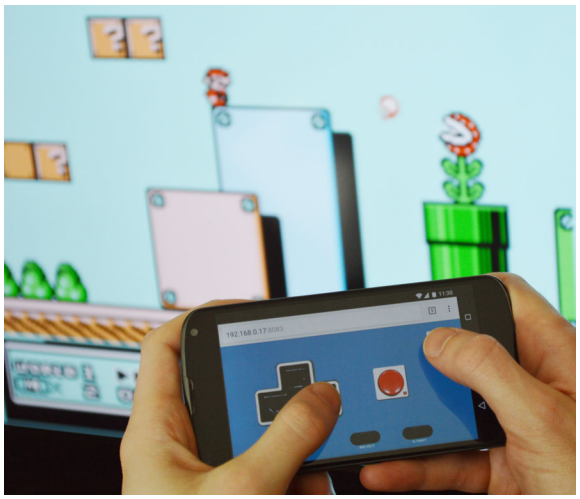
To allow for bidirectional communication, we considered a feedback mechanism. ATREUS provides different types of feedback messages such as a *TextMessage* for transmitting simple text (for example, 'thank you' after participation) or a *VibrationMessage* for triggering the device's vibration module in a specified timely pattern (for example, in case the player's game character was hit).

Such feedback messages can be either broadcasted to all connected clients or be targeted at a specific client. Due to this feedback mechanism, the application logic stays on the platform side and the mobile code remains lean. There is no need to duplicate logic parts for the mobile clients (for example, the calculation of a game score) since those can be easily synchronized by pushing important information through textual feedback messages.

Mobile Library

The *Mobile Library* is written in *JavaScript* and can be integrated in web pages to enable communication with the ATREUS platform. It contains convenient methods to connect to the platform, create control messages and send them to the platform. For example, a typical gamepad control would call the library's *createMobileAction* and *sendMobileAction* each time a button is pressed, to create a JSON-formatted text

[6]http://eclipse.org/jetty/



(a) An on-screen gamepad to control a platform game.

(b) Steering a racing car by tilting the device.

(c) A touch-enabled miniature view in full-screen mode.

(d) Interacting with the screen through the camera viewfinder.

Figure 3: Using the ATREUS framework, we prototyped four web-based remote controls as demonstrators: Gamepad (a), Driving Wheel (b), Mini Video (c), and Smart Lens (d).

string including central control parameters and submit it to the platform.

Further methods allow the data of built-in sensors, such as accelerometers, to be continuously streamed to the platform. For processing feedback messages from the platform, the mobile library offers a basic listener concept. The functions *addVibrationEventListener(func)* and *addSoundEventListener(func)* can be used to override the default behavior (vibrating and playing an audio file) with custom functions, *addTextEventListener(func)* specifies the handling of an incoming textual feedback.

DEMONSTRATORS

To demonstrate the basics of the described framework and show how to extend it with custom components for advanced remote controls, we built four functional prototypes (Figure 3) which were implemented as native apps in prior work (e.g., [3, 7, 20]).

Button-based Remote Controls

Realizing button-based remote controls, such as the on-screen gamepads (Figure 3a), is straightforward: when a touch is registered through the respective JavaScript calls, we create corresponding control messages through the mobile library (for example, one control message for each direction of a 4-way directional pad and two for two action buttons) and send them to the platform.

Driving Wheel

As an example for a remote control exploiting the built-in sensors of a smartphone, we realized a driving wheel for controlling a remote racing game by tilting the smartphone (see Figure 3b). A control message containing raw accelerometer values is generated and transmitted to the platform each time

the orientation of the device changes. At the platform, the values can be either translated to key presses when pre-defined tilt thresholds are reached (for example, the right cursor key is pressed when the tilt value is larger than 20 degrees) or, for a more realistic user experience, used to emulate continuous movements through a virtual joystick driver such as *vJoy*[7].

Mini Video

For realizing the *Mini Video* technique (a touch-sensitive miniature version of the remote screen content), we applied a separate WebSocket channel in addition to the one dedicated to exchanging remote control and feedback commands. This channel is purely used for distributing the content of the remote screen: A custom component at the platform continuously takes screenshots and encodes the binary image data into Base64 text strings which are sent over this second channel to connected mobile clients. This binary-to-text encoding step on the platform helps to reduce the computational efforts on the mobile devices since Base64-encoded images can be directly displayed using so-called *Data URIs*[8].

Depending on the required video quality, the size of the original screenshot can be reduced for quicker encoding and transmission. For example, for the *Mini Video* control depicted in Figure 3c the platform shrinks the original screenshot images by a factor of 0.5 while the mobile browser simply scales up these miniatures to full-screen using Cascading Style Sheets (CSS). The current prototype runs at 10 frames per second on a Nexus 4 device in an 802.11n WiFi network. Since the mobile web application is aware of the dimensions of the remote screen due to an initial welcome message, touches on the video can be mapped to actual screen coordinates and respective control commands created and sent to the platform.

Smart Lens

The most advanced remote control currently built with the ATREUS framework is the *Smart Lens* technique (Figure 3d). This recent input method enables interaction with a remote screen by targeting it through the camera viewfinder of the mobile device and pushing remote buttons by touching them on the mobile live video. It is typically realized by means of computer vision to match the camera frames with the current content of the remote screen and thus to map touches on the mobile display to according positions on the remote screen.

Camera access for web applications is enabled by the WebRTC (Web Real-Time Communication) API for voice calls and video chats. Instead of a continuous visual tracking of the screen content on the mobile device, we decided on a computationally less intensive implementation for our web-based version (cf. [4]). Besides sending the touch position in mobile display coordinates, we use a second (binary) WebSocket channel to transmit the current camera video frame as soon as the user touches the mobile display. Gaining access to the frame is currently only possible through a work-around: each camera frame needs to be drawn to a canvas acting as the viewfinder where it can be retrieved from in compressed JPG format as Data URI.

[7] http://vjoystick.sourceforge.net/

[8] https://developer.mozilla.org/en-US/docs/web/HTTP/data_URIs

After the transmission of the camera frame, a custom component at the platform side is responsible for creating a screenshot and comparing these two images. For this task, we use *BoofCV*[9], an open source Java library for real-time computer vision. For the Nexus 4 device connected in an 802.11n WiFi network, our current implementation takes about one second to translate a touch on the viewfinder (frame size 800x600) to the corresponding remote action.

LESSONS LEARNED

In this section, we summarize lessons learned during the development and testing of the ATREUS demonstrators. In particular we highlight challenges of providing web-based remote controls compared to native counterparts.

Browser support. We tested all demonstrators on iOS and Android, using the most recent mobile versions of Safari, Chrome, Firefox and Opera. All browsers provide the required methods to process touch events, access built-in sensors, display inline images, and establish WebSocket connections and thus supported the remote controls Gamepad, Driving Wheel, and Mini Video well.

The Smart Lens control requires access to the smartphone's camera. While this is fully supported by Chrome and Opera, the mobile Safari browser currently does not provide the HTML5 functions to infer information about built-in cameras and access them. Firefox Mobile is able to open the camera and show its video stream. However, in our implementation it fails to access single video frames. Even when browsers provide WebRTC support, developers should be aware of varying behaviors on different browsers. For example, Opera Mobile shows a menu for choosing between the front and the back camera while in many browsers the front camera (video telephony as the major use cases) is the default camera.

Permissions. One peculiarity of web applications are browser-initiated permission prompts. In contrast to native apps, permissions for security- or privacy-relevant features such as camera access can not be granted once during installation but need to be confirmed by the users each time they use the application. Similarly, modern mobile web browsers support switching to full-screen mode (i.e., hiding the browser's address bar) to enable a more native app-like experience, yet prompt a permission request.

Furthermore, a web application can currently only switch to full-screen mode after an explicit interaction by the user such as a button press. Thus, starting a remote control in full-screen by default is not possible, yet not a crucial requirement.

Placement of Controls. While for a native application soft buttons for system operations (such as the "back" button in Android) can be disabled, this is not possible for a web application. During fast-paced games, players may hit system buttons by mistake and close the browser, for example. Thus, buttons of a web-based remote control should not only conform to well-known minimum sizes for touch elements but should also not be placed close to system buttons to avoid unintentional hits.

[9] http://boofcv.org/

Screen Rotation. For tilt-based remote controls, designers need to consider the mobile platform's default screen rotation settings. Expansive gestures (with the Driving Wheel control, for example) may trigger an automated screen re-arrangement from landscape to portrait orientation. Consequently, potential on-screen buttons appear at different screen positions, making them more difficult to hit.

Hence, tilt-based controls should either not use touch elements or screen re-orientations should be detected and managed accordingly (by cleverly relocating touch elements, for example).

EVALUATION

To gain first insights on how users perceive these web-based remote controls, we conducted an informal user study with 15 users in our institution's user experience lab. We invited participants with diverse educational backgrounds and asked them to use the aforementioned prototypes to control a simple game on the remote screen for about 15 minutes. Finally, they were asked about their opinions on these web applications in a qualitative interview afterwards.

From a *performance* perspective, we found all our remote controls to run smoothly, leading to participants did not complain about reaction times. Furthermore, we were interested in the participants' *preference* if given the choice to use a web-based control versus a native control application. Eight out of 15 participants stated to not care about the underlying technology. For example P5 stated *"I don't care about the technology as long as it works."* whereas P11 mentioned that *"If there are no differences in functionality or costs, it's all the same to me."*. Two participants stated to prefer such web applications. Participants justified their decision by statements such as *"No download, no installation, that's convenient."* (P12) and *"The web app doesn't need any storage space."* (P3).

One participant with experience in software development raised trustworthiness as an additional advantage of web apps over native applications and stated *"I would rather trust in a web application since I can browse its source code."* (P2). Finally, five participants stated to prefer native counterparts. Statements included *"I prefer a native app because I have to install it only once. A browser-based application needs to be downloaded each time I want to use it."* (P5) and *"I rather like a native app because it does not need a constant data connection."* (P6).

The last comment hints at a potential challenge: users may associate costs (i.e., the need for an Internet connection) with using a browser-based control compared to using a native app. In fact, both types could be implemented in a way such that no Internet connection is required at all, for example by using a local wireless network.

As solution, for an interactive display application, a WiFi network (without Internet connectivity) could be provided that users can connect to. This provides a convenient way of accessing the remote control, as the router could immediately direct to the URL for the remote control.

CONCLUSIONS AND OUTLOOK

In this paper, we investigated current opportunities and drawbacks of realizing web-based remote controls for smartphones. We introduced ATREUS, an open source framework for realizing advanced browser-based remote controls for smartphones to control interactive applications on remote screens. Based upon the basic framework, we presented four demonstrators including sophisticated remote controls such as the Mini Video and the Smart Lens technique which previously were realized as native apps. Experiments with these prototypes and an informal study yielded useful overall insights on the design of web-based remote controls for smartphones.

After these first proof-of-concept prototypes, we see the optimization of the visual remote controls as promising future work. Our current Mini Video implementation could be enhanced by integrating efficient web-suitable codecs for enabling smoother video streaming. Recent advancements in web-based computer vision (e.g., *tracking.js*[10]) could enable the continuous tracking of the remote content by the Smart Lens and help to create more seamless interactions and realize more complex use cases in the future.

After our pilot study with users, we plan to invite developers to gain feedback on the API design and assess its usability. Further, we expect to receive suggestions for potential extensions in brainstorming sessions.

ACKNOWLEDGMENTS

This work has been carried out within the project ATREUS financed by the *netidee* initiative of the Internet Foundation Austria (IPA).

REFERENCES

1. Florian Alt, Thomas Kubitza, Dominik Bial, Firas Zaidan, Markus Ortel, Björn Zurmaar, Tim Lewen, Alireza Sahami Shirazi, and Albrecht Schmidt. 2011. Digifieds: Insights into Deploying Digital Public Notice Areas in the Wild. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia (MUM '11)*. ACM, New York, NY, USA, 165–174. DOI: **http://dx.doi.org/10.1145/2107596.2107618**
2. Florian Alt, Alireza Sahami Shirazi, Thomas Kubitza, and Albrecht Schmidt. 2013. Interaction techniques for creating and exchanging content with public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1709–1718. DOI: **http://dx.doi.org/10.1145/2470654.2466226**
3. Matthias Baldauf, Peter Fröhlich, Jasmin Buchta, and Theresa Stürmer. 2013. From Touchpad to Smart Lens: A Comparative Study on Smartphone Interaction with Public Displays. *IJMHCI* 5, 2 (2013), 1–20. DOI: **http://dx.doi.org/10.4018/jmhci.2013040101**
4. Matthias Baldauf, Peter Fröhlich, and Peter Reichl. 2010. Touching the untouchables: Vision-based

[10] http://trackingjs.com/

real-time interaction with public displays through mobile touchscreen devices. In *Adj. Proceedings of the 8th International Conference on Pervasive Computing (Pervasive10)*. Helsinki, Finland.

5. Matthias Baldauf, Stefan Suette, Peter Fröhlich, and Ulrich Lehner. 2014. Interactive Opinion Polls on Public Displays: Studying Privacy Requirements in the Wild. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI '14)*. ACM, New York, NY, USA, 495–500. DOI: http://dx.doi.org/10.1145/2628363.2634222
6. Rafael Ballagas, Jan Borchers, Michael Rohs, and Jennifer G. Sheridan. 2006. The Smart Phone: A Ubiquitous Input Device. *IEEE Pervasive Computing* 5, 1 (Jan. 2006), 70–77. DOI: http://dx.doi.org/10.1109/MPRV.2006.18
7. Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch. 2010. Touch Projector: Mobile Interaction Through Video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2287–2296. DOI: http://dx.doi.org/10.1145/1753326.1753671
8. Jorge C. S. Cardoso and Maria Barreira. 2014. A Web-based Toolkit for Remote Direct Manipulation Interaction with Public Displays via Smartphones. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MOBIQUITOUS '14)*. 357–358. DOI:http://dx.doi.org/10.4108/icst.mobiquitous.2014.258067
9. Jorge C. S. Cardoso and Rui José. 2012. Creating Web-based Interactive Public Display Applications with the PuReWidgets Toolkit. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia (MUM '12)*. ACM, New York, NY, USA, Article 55, 4 pages. DOI: http://dx.doi.org/10.1145/2406367.2406434
10. Sarah Clinch. 2013. Smartphones and Pervasive Public Displays. *IEEE Pervasive Computing* 12, 1 (2013), 92–95. DOI: http://dx.doi.org/10.1109/MPRV.2013.16
11. Nigel Davies, Sarah Clinch, and Florian Alt. 2014. *Pervasive Displays: Understanding the Future of Digital Signage*. Morgan & Claypool Publishers. DOI:http://dx.doi.org/10.2200/S00558ED1V01Y201312MPC011
12. Tilman Dingler, Tobias Bagg, Yves Grau, Niels Henze, and Albrecht Schmidt. 2015. uCanvas: A Web Framework for Spontaneous Smartphone Interaction with Ubiquitous Displays. In *Human-Computer Interaction–INTERACT 2015*. Springer, 402–409. DOI: http://dx.doi.org/10.1007/978-3-319-22698-9_27
13. Aiman Erbad, Michael Blackstock, Adrian Friday, Rodger Lea, and Jalal Al-Muhtadi. 2008. MAGIC Broker: A Middleware Toolkit for Interactive Public Displays. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM '08)*. IEEE Computer Society, Washington, DC, USA, 509–514. DOI: http://dx.doi.org/10.1109/PERCOM.2008.109
14. Tommi Heikkinen, Jorge Goncalves, Vassilis Kostakos, Ivan Elhart, and Timo Ojala. 2014. Tandem Browsing Toolkit: Distributed Multi-Display Interfaces with Web Technologies. In *Proceedings of The International Symposium on Pervasive Displays (PerDis '14)*. ACM, New York, NY, USA, Article 142, 6 pages. DOI: http://dx.doi.org/10.1145/2611009.2611026
15. Tomas Linden, Tommi Heikkinen, Timo Ojala, Hannu Kukka, and Marko Jurmu. 2010. Web-based Framework for Spatiotemporal Screen Real Estate Management of Interactive Public Displays. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 1277–1280. DOI: http://dx.doi.org/10.1145/1772690.1772901
16. Krzysztof Pietroszek, James R. Wallace, and Edward Lank. 2015. Tiltcasting: 3D Interaction on Large Displays Using a Mobile Device. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 57–62. DOI: http://dx.doi.org/10.1145/2807442.2807471
17. Stefan Schneegass and Florian Alt. 2014. SenScreen: A Toolkit for Supporting Sensor-enabled Multi-Display Networks. In *Proceedings of The International Symposium on Pervasive Displays (PerDis '14)*. ACM, New York, NY, USA, Article 92, 6 pages. DOI: http://dx.doi.org/10.1145/2611009.2611017
18. James She, Jon Crowcroft, Hao Fu, and Pin-Han Ho. 2013. Smart Signage: An Interactive Signage System with Multiple Displays. In *Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing (GREENCOM-ITHINGS-CPSCOM '13)*. IEEE Computer Society, Washington, DC, USA, 737–742. DOI:http://dx.doi.org/10.1109/GreenCom-iThings-CPSCom.2013.133
19. Alireza Sahami Shirazi, Christian Winkler, and Albrecht Schmidt. 2009. Flashlight interaction: a study on mobile phone interaction techniques with large displays. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09)*. ACM, New York, NY, USA, Article 93, 2 pages. DOI: http://dx.doi.org/10.1145/1613858.1613965
20. Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. 2008. Using a mobile phone as a Wii-like controller for playing games on a large public display. *International Journal of Computer Games Technology* 2008 (2008). DOI: http://dx.doi.org/10.1155/2008/539078