RED: a framework for prototyping multi-display applications using web technologies.

Roberto Calderon, Michael Blackstock, Rodger Lea, Sidney Fels
Human Communication Technologies Laboratory
University of British Columbia
roberto@alumni.ubc.ca, mblackst@magic.ubc.ca, rodgerl@ece.ubc.ca

Andre de Oliveira Bueno, Junia Anacleto
Advanced Interaction Laboratory
Department of Computer Science. Federal University of São Carlos.
andre.obueno@dc.ufscar.br, junia@dc.ufscar.br

ABSTRACT

We present the Really Easy Displays framework (RED), a web-based platform to facilitate spontaneous interaction between devices and applications. RED provides a single abstraction for content and interaction between display types, data streams and interaction modalities, and allows developers to create multi-display applications by enabling the sharing of web document object models (DOMs) across displays. We present lessons learned from using RED in our own research, hands-on workshops with developers and interviews with long-term developers over the course of a year. We provide initial evidence that the use of web-technologies in a framework like RED can mitigate some barriers encountered in by multi-display interaction scenarios, and we propose future work to improve RED.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Software libraries, Modules and interfaces*

1. INTRODUCTION

There has been significant research activity to investigate issues related to pervasive displays. These have included deployment considerations [6, 19], technologies [16], their role in public places [4] and communities [15]. Although displays are pervasive components of our world, developing and deploying applications that provide spontaneous interaction in multi-display scenarios is often challenging. Previous research has highlighted a number of issues including: the need to synchronize data sources and support interaction flows across multiple displays [8]; minimizing the existing barriers to initial interaction with multi-display scenarios [7] and simplifying spatial and temporal management of applications [13].

In our own research we have explored issues related to infrastructure for multi-display scenarios, questions regarding how users engage with public displays [20, 2], and design issues when large screens are paired with personal displays [12]. Throughout this research we have collected evidence that the development and deployment of multi-display applications is cumbersome, particularly for research scenarios that require rapid prototyping. Our observations align with those of other researchers, namely that spontaneous interaction with public displays is difficult and requires specialized software or hardware [8, 1]; deployment and maintenance is difficult and time-consuming [13]; and many of the elements that form multi-display applications, like data streams and interaction events, have differing attributes and relations that make them difficult to inter-connect [7].

To tackle these issues we have developed the Really Easy Displays (RED) framework. RED leverages web technologies to allow interoperability between web page objects across displays. These elements can include text, images, video, and interaction events. This approach has the benefit of simplifying the development of multi-display applications, but can be limiting for applications outside the common scenario of a mobile device interacting with situated displays. We present lessons learned while developing the framework.

2. BACKGROUND

As indicated in the introduction, there has been significant research into pervasive displays. Our focus, application frameworks for multi-screen display networks, has also seen significant investigation which, for the purpose of this paper, we group into *web centric* and *non web centric* approaches, highlighting the focus in the former on using web technologies.

Non web centric work on supporting the development of multi-display applications includes frameworks that investigate abstractions of virtual and physical objects, and frameworks that leverage off-the-shelf software previously installed in user's devices. Rukzio et al. [18] developed a framework based on single abstractions between devices to allow mobile devices to connect with physical objects, displays, places or other uses. Biehl et al. [1] proposed a framework to support collaboration in multi-display environments by allowing users to place off-the-shelf applications on shared displays. Clinch et al. [5] have explored software players for media scheduling and playback on networks of public displays, focusing on openness, extensibility and resilience.

Research on the use of web technologies to support multi-display applications includes investigations on leveraging web objects to develop applications using Internet events, and supporting spatial and temporal management of multi-display applications. Han et al. [8] investigated extending existing HTML mark-up with a unified XML to represent web documents and physical objects. Johanson et al. [10] explored the use of event managers to synchronize web page states across devices, allowing web content to be moved between devices with ease.

Ferscha et al. [7] proposed a framework called WebWall to allow multi-user interaction with public displays. Their framework links together different protocols (e.g. HTTP, email. SMS, MMS, etc.) to simplify the development of applications that link together different data sources. This is achieved by abstracting data as separate from displays and making it web-accessible. Linden et al. [13] propose a web-based framework to manage applications for interactive public displays in both spatial and temporal dimensions. Cardoso et al. [3] propose a framework to develop multi-display applications as web applications through the use of widgets such as list boxes, upload and download buttons, and a "check-in" button.

Although previous research has explored the use of web technologies to assemble multi-display applications (Johanson, Ferscha), these frameworks often require the use of native device applications. Our framework adopts a web-centric approach that eliminates the need to manage differences between display types and hardware. We follow the steps taken by Rukzio and propose an abstraction of data sources and interaction modalities that could be implemented in a web centric infrastructure. Research by Linden et al. gives us the confidence that web applications are well suited for multi-display applications, providing ease of development and management. We build on lessons learned by Cardoso et al., by exploring the possibility of enabling the sharing of content and interaction events between displays using standard web technologies, particularly standard DOM objects of web pages. In comparison to other frameworks RED provides a single development, deployment and application management solution for interactive multi-display applications. RED leverages the pervasive nature of web browsers and allows applications to be developed using only widely-used web technologies (HTML, CSS and Javascript). The framework allows web document's DOM objects to share events across displays and access other non-web data sources (e.g. physical things, gesture sensors) with ease.

3. REQUIREMENTS FOR RED

The proliferation of public displays and personal devices such as cell phones or tablets offers a unique and growing environment for multi-display interaction in the real world. The following scenario highlights some of the challenges that public-facing multi-display applications face:

> *Helen is a regular visitor of a local coffee shop that fashions a set of large wall-mounted LCD displays often used to advertise local events and community activities. Today, the displays show a set of multi-media posts that coffee shop visitors have made regarding the question "Should we discontinue the strawberry-flavoured latte?". Intrigued, Helen places her tablet on the table in front of her. The device automatically visits a web-application that allows her to access content posted by other customers and broadcasts her opinions on the topic. She drafts a message on her tablet and with a gesture broadcasts her answer to the large displays. As her post joins the stream of content that shapes a lively discussion, her coffee mug begins to change colour (through a set of hidden LEDs) rewarding her for her participation and discreetly announcing she has participated in a common effort.*

In the above scenario, Helen is able to interact with many displays without the need to install custom software. Moreover, even though data comes from different sources and is distributed across many displays, they work together to form a coherent experience where interaction flows naturally. In our own research [12] we have found that two of the most critical challenges for building applications in public multi-display scenarios like this one are firstly, spontaneous interaction with situated displays allowing for bystanders to interact directly with displays, and secondly, bringing together different types of displays (e.g. mobile phones, large screens, ambient displays, sensors and actuators) with differing capabilities and interaction modalities [11]. We propose two requirements to a framework for multi-display applications:

1. Facilitating spontaneous interaction between devices and applications without the need for specialized software to be installed in each device.

2. Providing a simple abstraction for distributing content and interaction events that can span multiple display types, data streams and interaction modalities (touch, gesture, point-and-click).

4. RED ARCHITECTURE

The Really Easy Displays Framework (RED) was designed to address the above requirements and consists of three components (1) event management, (2) application virtualization, and (3) data abstraction:

- Object Sharing and Eventing: Consists of an application API providing tools to share, and synchronize web DOM (Document Object Model) objects across displays through events,

- Application Virtualization: Is composed of an *application container* providing application discovery, application management (deployment and queuing), and basic functionalities (user accounts, application selection, public message boards and private messages) for multi-display applications, and

- Object Abstraction: Supporting infrastructure providing a simplified abstraction for differing protocols and data sources (*Thing Broker*).

The key idea behind the RED Framework is the sharing of web document object model (DOM) elements across displays. RED leverages the ubiquity of web browsers on contemporary display platforms (e.g. modern mobile phones, tablets, and televisions) and the well supported access to

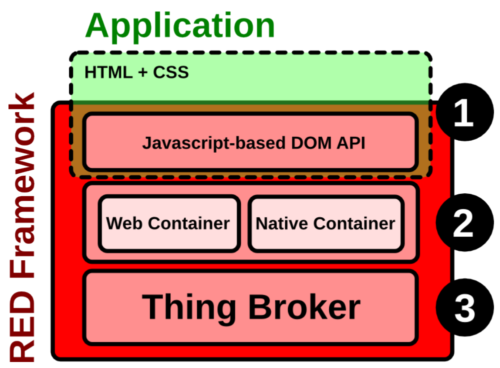

Figure 1: Architecture of the RED Framework. (1) Javascript-based DOM API allowing web document objects to produce and consume events. (2) Application container allowing interaction with displays without custom software and instantiating application threads. (3) Supporting infrastructure providing abstraction for data and protocols.

RESTful APIs using such devices. The framework was designed under the premise that modern web-browsers can be found on most contemporary displays. This makes it possible to model multi-display applications as a collection of web page objects whose state can be manipulated across displays.

4.1 Object sharing and eventing: DOM API

Developers using RED need only to interact with the RED DOM API, a RESTful Javascript API, which allows the sharing of interaction events using a RED backend server. When used in the context of the framework, the RED DOM API allows developers to extend common DOM elements to be shared transparently between interacting displays. These extended DOM elements can handle common browser manipulations (append, prepend and remove) and browser events (click, tap-on, focus, resize, etc.) across displays. Finally, custom events can be sent and received by these DOM elements. With this approach, applications only need to be aware of establishing RED DOM objects as the framework abstracts the underlying complexity of event management in a multi-display scenario.

4.2 Application Virtualization: Container

The RED Application Container is responsible for managing RED 'applications' and providing a set of interaction services for a particular place, scenario or network of displays. The container serves a variety of purposes. Firstly, it provides a set of mechanisms to thread data or interaction events specific to an application, place or interaction flow. When an application is deployed using the RED Application Container, the DOM API creates threads for each DOM object using web-sessions and cookies, ensuring that events are delivered to the correct thread. This creates "virtual" versions of a RED application that can be deployed across a selected group of displays. Secondly, the container provides a set of generic end user services such as messaging and notifications, basic social functionality (place *checkins*) and other services which we have found to be common to many pervasive display applications. In addition to this, the container provides application hosting for developers without access to a hosting service, logging capabilities for usage analysis and optional access to user information (e.g. profile picture, name, interests, etc.) using "Facebook-Login". To use these functionalities and access available applications users typically scan a QR code or NFC tag, or visit a short URL. This can be done via a RED container native to a display (Android or iOS) or by using a web browser.

4.3 Object Abstraction: Thing Broker

The Thing Broker is a platform that has grown out of our research on systems and infrastructure for the Internet of Things (IoT) [17]. Figure 3 presents the data model of the Thing Broker: (1) meta-data associated with object to describe them with human-readable properties, (2) streams of data called *events* consisting of typed name/value pairs that can be numbers, strings, objects or arrays, and (3) associated MIME-type resources (unstructured text, HTML, photos, video or audio clips). Using the RED API, developers map DOM objects to Thing Broker "things", leveraging its ability to share state and meta data and interaction events between shared DOM objects on different displays. By using the Thing Broker, RED applications have access to data sources outside web documents, like sensors and actuators (e.g. pressure sensors, relays, accelerometers) or gesture interaction from devices such as the Kinect.

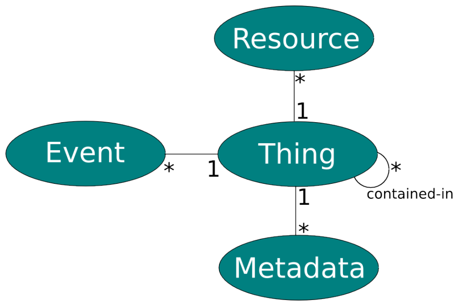

Figure 3: Abstraction of data using the concept of "thing" used in the Thing Broker.

5. EXPERIENCE USING RED

We have used RED both internally to our research group and with a number of external groups. Internally we have developed a range of applications of differing complexity to explore different pervasive display scenarios, In addition, we have run workshops for developers of multi-display applications who were new to the framework, and have carried out informal interviews and questionnaires with developers outside of our research group that have used RED.

5.1 Application development

Over the course of 18 months our research laboratory has developed a wide range of applications using RED. These applications include:

Information broadcasting applications: a presentation application that allows users to browse through presentation slides using their mobile phones; and a photo gallery that allowed people to broadcast images from their mobile devices on a public display.

Collaborative applications: a competitive memory game where a large display is used to display a pattern that users

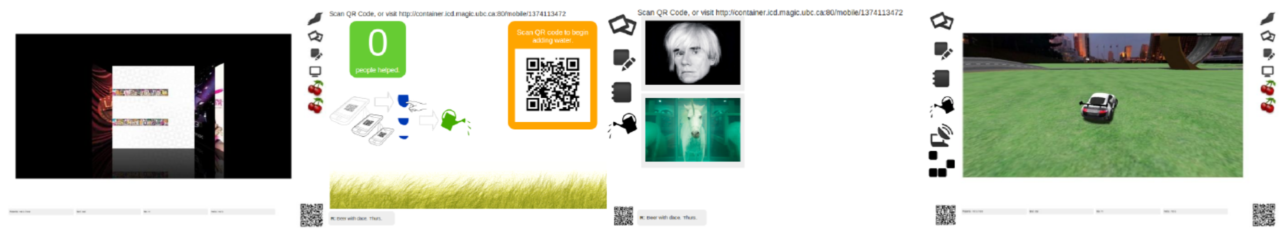

Figure 2: The Application Container on a large screen showing four RED applications: a music jukebox, a garden application, an image gallery, and a race simulation game. Users wanting to interact with these applications can scan the QR-code shown to interact with applications.

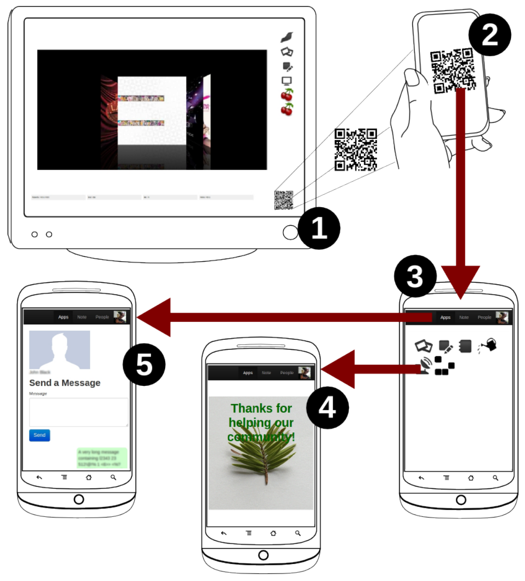

Figure 4: (1) A large screen displays a QR Code and available applications. (2) Users can scan this QR code using their mobile devices to (3) interact with the large display and select applications. (4) Selecting applications is done by tapping on icons. (5) Posting notes to message boards and sending private messages to other people are always available to users.

match on their mobile phones; and a music jukebox that uses a large situated display to play music that people upload and control using their mobile phones.

Mobile device applications: a visualization of private messages sent by people using their mobile phones within the RED container showing social interactions between members of a community; a situated large display showing a virtual garden where plants are nurtured when users *checkin* into a place through their mobile devices; and a car racing simulation game where a WebGL [14] car on a large display is controlled by users tilting their mobile phones.

Application to manipulate data: An application to create and edit medical records was developed leveraging a large display

5.2 Workshops and Developers

We two conducted two-hour workshops aimed at introducing the RED framework to developers through creating a sample application or working with them to rapidly prototype an application of their own. An exit questionnaire was used to understand participants' previous knowledge of dynamic web pages and their experience using RED. In addition to these workshops we conducted informal interviews with developers that used RED on their own. Interviews aimed at gaining a deeper understanding of the experience that developers had in using RED over a longer period of time. Participants included developers at three Canadian Universities and one Brazilian University. In total, 11 developers participated in our workshops: 4 Canadian and 7 Brazilian. 63% of participants said that they had previous experience writing HTML documents, and 37% said that they had moderate experience with dynamic web documents. Fourteen (14) developers that used RED over a longer period of time where interviewed. Only 36% of participants said that they had previous experience using Javascript, while the rest had never developed dynamic web documents using Javascript.

6. LESSONS LEARNED

In the above experiences using RED we learned that pervasive display applications are only a small part of a growing ecosystem of devices and data sources that extend interaction events available in modern web browsers. Frameworks that aim at supporting multi-display applications need to take into account such complexities and provide affordances that are appropriate for such scenarios. Key lessons learned are:

Multi-display applications are more than mobile devices interacting with large displays. We originally designed RED for the predicted scenario of many mobile displays interacting with a single large display. Applications needing mobile-to-mobile interaction, or a single mobile device interacting with many co-located large displays found RED limiting. For example, while developing the memory game and garden applications, we had to customize how DOM events were produced to allow mobile devices to interact with each other without the need for a large display as intermediary.

Common Web browser interactions are not sufficient for multi-display applications. Adapting web interaction events common in modern web browsers for multi-display applications can be limiting. Whilst flexible, the palette of data sources and interaction events provided by RED was not sufficiently rich. We observed that developers of multi-display applications often require support for user interactions that lie outside the common capabilities of a web browser; for example hand gestures or positioning of a display in relation to a user's body. For example, in the WebGL racing game application developed in our laboratory we had to use third party libraries to read accelerometer data and transform it to RED events through a hidden DOM object, while in the garden and visualization applications we had to implement custom types of events, like "actively interacting with a display" or "interacting with another mobile display".

Open system affordances can hinder usage. In our original design of RED we exposed the functionality of the lower layers of the framework, like thing meta-data and event production/consumption . We observed that some developers adopted the affordance of "things" offered by RED in ways we did not originally expect. For example, developers made use of the Thing Broker's state and meta-data storage capabilities to store application domain models such as user profiles and messages, effectively using RED as a database rather than a framework to synchronize DOM states across displays. Specifically, in the medical records application, developers used the Thing Broker to store and manipulate persistent data needed by their application, bypassing the Application Container. Although this flexibility was initially useful it interfered with the expected functioning of RED applications. For example, when bypassing the Application Container applications lost their ability to be efficiently virtualized and managed by developers.

The complexity of real-world deployments should not be underestimated. We observed that deployment of multi-display applications in real world scenarios is more complex than we anticipated. In our workshops and interviews with developers we found that our approach to building applications using a cloud accessible service (Thing Broker) could be limiting, rather than empowering for some scenarios. For example, in a deployment in a Brazilian hospital, permanent and reliable connectivity to the Internet was not available. Because of this, developers used local or private networks to host RED and its associated infrastructure. This resulted in considerable efforts towards managing several instances of RED, rather than focusing on building multi-display applications. It is hard for a framework like RED to predict all complications that might arise in real world deployments, but a more careful and systematic approach, like the one proposed by Storz et al. [19], can help mitigate such complexities.

Displays are only a small part of an UbiComp ecosystem. With the increased availability of consumer devices and Internet based services multi-display applications often extend onto a growing ecosystem of ubiquitous computing (e.g. social data, Internet-connected sensors and actuators, wearables). Developers using RED began using the term "things", used by some layers of the framework, to refer to a wide selection of entities outside of what we originally thought composed multi-display applications: content, sensors, actuators, persistent data, social data. Often, developers would use the lower layers of the framework (Thing Broker) or manipulate DOM objects to represent non-display entities.

7. FUTURE OF RED

Based on these findings we have re-architectured some parts of the RED framework, have worked on expanding the palette of data sources and interaction modalities, and have been experimenting with application scenarios outside our initial scenario of mobile displays interacting with a single situated display.

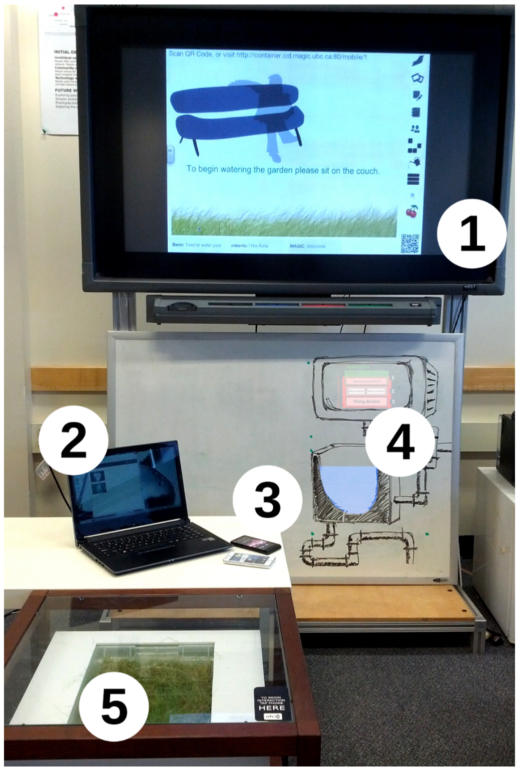

Figure 5: Future work on using RED: an application that utilizes a (1) large display, a (2) personal computer, a (3) mobile phone, (4) gesture-enabled projected displays and a (5) physical garden.

We have extended the selection of data sources of RED by incorporating a microcontroller API allowing developers to link the input and output state of microcontrollers' pin states (Arduino) to DOM objects in a website. Similarly, we have implemented a body-gestures gateway using a Microsoft Kinect that developers can utilize to link a palette of social gesture events (handshakes or fist bumps between people) to DOM states in RED applications.

Furthermore, we are experimenting with deploying RED applications across new types of displays, including gesture-enabled projected displays like the UbiDisplays framework [9]. Some of the applications we have developed in our laboratory using the extended and improved RED framework include: a memory game that utilizes large displays, mobile devices and projected displays (UbiDisplays framework); and a physical garden whose water pump can be actuated by means of touching virtual water projected on a wall and interacting with pressure sensors on furniture.

8. CONCLUSION

The Really Easy Displays (RED) framework aims to simplify the development, deployment and management of multi-display applications. Toward this goal, RED was designed to address the challenges of (1) supporting and facilitating spontaneous interaction between devices and applications without the need for specialized software to be installed in each device, and (2) using a single abstraction that can represent content and interaction that can span multiple display types, data streams and interaction modalities. RED allows the modelling of applications as interconnected web (DOM) objects, greatly simplifying the development of multi-display prototype applications.

We documented our experience of using RED by developing a wide range of applications in our own research, conducting workshops with developers and interviewing long term users of the framework. We found that RED's web-centric approach has proven successful at mitigating some of the most important barriers posed by multi-display scenarios. Namely, spontaneous interaction with public displays is complex and requires specialized software or hardware; deployment and maintenance is difficult and time-consuming; and many of the elements that form multi-display applications, like data streams and interaction events, have differing attributes and relations that make them difficult to inter-connect.

We learned that displays are only a small part of a growing UbiComp ecosystem. Specifically, multi-display applications are more than mobile devices interacting with large displays. With the increased availability of consumer devices and Internet based services multi-display applications often extend across a growing ecosystem of ubiquitous computing (e.g. social data, Internet-connected sensors and actuators, wearables). As a result, even cutting-edge web browser interactions (e.g. touch, swipe, geolocation, accelerometer data) are not sufficient for multi-display applications needing to extend onto this ecosystem. Extending such affordances requires further research because, as we learned, open affordances can hinder usage and adoption of new frameworks.

Based on our experience to date, we believe that RED and similar frameworks need to (a) support complex scenarios of multi-display applications where many types of devices interact with displays, and (b) provide a rich palette of data sources and interaction modalities. Future work on RED includes extending the current palette of interaction modalities and data sources and experimenting with different application scenarios and display types.

9. REFERENCES

[1] J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski. Impromptu: A new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. CHI '08, pages 939–948, New York, NY, USA, 2008. ACM.

[2] M. Blackstock, N. Kaviani, R. Lea, and A. Friday. Magic broker 2: An open and extensible platform for the internet of things. pages 1 –8, 2010.

[3] J. C. S. Cardoso and R. José. Evaluation of a programming toolkit for interactive public display applications. MUM '13, pages 6:1–6:10, New York, NY, USA, 2013. ACM.

[4] E. F. Churchill, L. Nelson, L. Denoue, J. Helfman, and P. Murphy. Sharing multimedia content with interactive public displays: a case study. DIS '04, pages 7–16, New York, NY, USA, 2004. ACM.

[5] S. Clinch, N. Davies, A. Friday, and G. Clinch. Yarely: A software player for open pervasive display networks. PerDis '13, pages 25–30, New York, NY, USA, 2013. ACM.

[6] N. Davies, M. Langheinrich, R. Jose, and A. Schmidt. Open display networks: A communications medium for the 21st century. *Computer*, 45(5):58–64, May 2012.

[7] A. Ferscha and S. Vogl. Pervasive web access via public communication walls. Pervasive '02, pages 84–97, London, UK, UK, 2002. Springer-Verlag.

[8] R. Han, V. Perret, and M. Naghshineh. Websplitter: A unified xml framework for multi-device collaborative web browsing. CSCW '00, pages 221–230, New York, NY, USA, 2000. ACM.

[9] J. Hardy and J. Alexander. Toolkit support for interactive projected displays. MUM '12, pages 42:1–42:10, New York, NY, USA, 2012. ACM.

[10] B. Johanson, S. Ponnekanti, C. Sengupta, and A. Fox. Multibrowsing: Moving web content across multiple displays. volume 2201 of *Lecture Notes in Computer Science*, pages 346–353. Springer Berlin Heidelberg, 2001.

[11] N. Kaviani, M. Finke, S. Fels, R. Lea, and H. Wang. What goes where?: designing interactive large public display applications for mobile device interaction. ICIMCS '09, pages 129–138, New York, NY, USA, 2009. ACM.

[12] N. Kaviani, M. Finke, R. Lea, and S. Fels. Investigating a design space for multidevice environments. volume 28, pages 722–729, 2012.

[13] T. Linden, T. Heikkinen, T. Ojala, H. Kukka, and M. Jurmu. Web-based framework for spatiotemporal screen real estate management of interactive public displays. WWW '10, pages 1277–1280, New York, NY, USA, 2010. ACM.

[14] K. Matsuda and R. Lea. *WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL.* Addison Wesley, 2013.

[15] N. Memarovic, M. Langheinrich, F. Alt, I. Elhart, S. Hosio, and E. Rubegni. Using public displays to stimulate passive engagement, active engagement, and discovery in public spaces. MAB '12, pages 55–64, New York, NY, USA, 2012. ACM.

[16] J. Müller, F. Alt, D. Michelis, and A. Schmidt. Requirements and design space for interactive public displays. MM '10, pages 1285–1294, New York, NY, USA, 2010. ACM.

[17] R. A. Perez de Almeida, M. Blackstock, R. Lea, R. Calderon, A. F. do Prado, and H. C. Guardia. Thing broker: A twitter for things. UbiComp '13 Adjunct, pages 1545–1554, New York, NY, USA, 2013. ACM.

[18] E. Rukzio, S. Wetzstein, and A. Schmidt. A framework for mobile interactions with the physical world. WPMC'05, 2005.

[19] O. Storz, A. Friday, N. Davies, J. Finney, C. Sas, and J. Sheridan. Public ubiquitous computing systems: Lessons from the e-campus display deployments. *IEEE Pervasive Computing*, 5(3):40–47, 2006.

[20] A. Tang, M. Finke, M. Blackstock, R. Leung, M. Deutscher, and R. Lea. Designing for bystanders: reflections on building a public digital forum. CHI '08, pages 879–882, New York, NY, USA, 2008. ACM.