Enriched Multiscreen TV Experience toolkit

Daniel Ockeloen
Noterik BV
Prins Hendrikkade 120
Amsterdam, The Netherlands
+31 (0)20 592 99 66
daniel@noterik.nl

Kati Hyyppä
Noterik BV
Prins Hendrikkade 120
Amsterdam, The Netherlands
+31 (0)20 592 99 66
kati@noterik.nl

Pieter van Leeuwen
Noterik BV
Prins Hendrikkade 120
Amsterdam, The Netherlands
+31 (0)20 592 99 66
p.vanleeuwen@noterik.nl

ABSTRACT
We present a *multiscreen toolkit* that is being developed in the LinkedTV project. The project researches and develops solutions for seamless integration of television and Web content, providing an enriched audiovisual experience. The multiscreen toolkit enables using devices like tablets as a 2nd screen in combination with television. The toolkit not only extends the interactive capabilities of television, but also enables versatile prototyping of multiscreen applications using HTML5 technologies. Our demonstration consists of a 2nd screen application implemented with the multiscreen toolkit which supports (1) viewing of time-based and spatial enrichments related to a TV program on mobile devices and (2) social interaction between viewers while watching a program.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: *Prototyping, Interaction styles, Graphical user interfaces;* H.5.1 [**Multimedia Information Systems**]: *Video;* C.2.4 [**Distributed Systems**]: *Distributed applications.*

General Terms
Design, Human Factors, Experimentation

Keywords
Interactive television, multiscreen, second screen, social video, toolkit

1. INTRODUCTION
Smart TV sets and mobile devices such as tablets have become part of our everyday media consumption, and multitasking with different devices is common.[1] Web technologies such as HTML5 are also used increasingly in providing new ways to engage with television content.[2] The LinkedTV project researches and develops new ways to integrate television and Web content seamlessly, providing enriched and interactive television experiences (http://www.linkedtv.eu). In the LinkedTV platform, enrichments related to a program are generated based on automatic video analyses and harvesting of related Web content. These temporally and spatially labeled enrichments are integrated together with interactive and social features in a non-intrusive user interface using a 2nd screen solution.

Our demonstration shows ongoing technical developments related to the interface design, which are continuation to prior LinkedTV work.[1] We present an HTML5 based multiscreen toolkit developed by Noterik (http://www.noterik.nl), which enables development of interactive television applications that operate across devices and screens. As an example, we show a 2nd screen application built with the toolkit, which enables both viewing of enriched television content as well as social interactions between viewers. The television content used in our demonstration is a news program called "rbb AKTUELL" by one the LinkedTV consortium members, Rundfunk Berlin-Brandenburg (http://www.rbb-online.de).

The core themes related to our technical demonstration, which is targeted to both designers and developers, include the following:

How to support presentation of enriched television content in non-intrusive ways using multiple screens, such as television and a touch screen tablet

How to support interaction between viewers with multiple screens

How to build a multiscreen toolkit, which enables flexible design and testing of multiscreen applications

2. DEMO SCENARIO
The idea of the LinkedTV platform is to automatically enrich television programs with various types of related information and media content from the Web. The program related information that is provided to the viewer should be relevant to the viewer's interests as well as to the type of program (e.g. news). The enrichments and interactive features should be also presented such that they do not intrude while watching the program. With these aspects in mind, we can depict a general scenario for our demonstration:

Laura starts to watch a television program and opens a 2nd screen application on her tablet. The application shows her personalized dashboard where she can join the program and obtain related information. As the program runs on the television, the enrichments related to it are displayed on the 2nd screen. Laura can explore them, or just watch the program, as the enrichments are also available for later viewing. She can also bookmark items related to the program, or share them with friends.

Figure 1 below elaborates the scenario further and gives an overview of some of the interactive features that are available in the 2nd screen application built with the multiscreen toolkit. A video runs on the main screen (top), while additional information is shown on a tablet (bottom left). The enrichments are show in layers (e.g. "People"), and items related to current video moment are highlighted. It is possible to view the related information in detail, bookmark it, share it, or to push it to the main screen using drag and drop (bottom right). The user can also adjust with settings what kind of information is shown, and on which screen it is shown (e.g. information layers can be turned on/off, and it can be adjusted how and where program related information or social items are displayed).

[1] See for example the following studies: http://razorfishoutlook.razorfish.com/articles/forgetmobile.aspx, http://www.google.com/think/research-studies/the-new-multi-screen-world-study.html

[2] See for example: http://www.hbbtv.org, http://dashif.org

Figure 1: Overview of the 2nd screen application built with the multiscreen toolkit.

3. INTENDED DEMO AUDIENCE

The demonstration is intended for concept/interaction designers and technical developers.

Concept/interaction designers: We are interested in showing the multiscreen toolkit to designers in the area of interactive television and social video applications. We would like to discuss possible further extensions to our toolkit, which enable flexible prototyping of concept and interface ideas.

Technical developers: We want to show the multiscreen toolkit to developers working on advanced HTML5 and multiscreen applications. We would like to discuss technical aspects of multiscreen application development as well as open sourcing of the multiscreen toolkit.

4. TECHNICAL DETAILS

The demonstration will show the 2nd screen application described above, which is one example of a multiscreen application that is being developed using the multiscreen toolkit. The aim of the toolkit is to allow rapid prototyping for multiscreen applications, where the developer can focus on implementation of his idea without having to deal with issues regarding, for example, synchronization or communication between the multiple screens.

The front-end of the demo is built for web browsers using HTML5, JavaScript and CSS technology in order to allow support for a broad variety of devices. Special attention is given to mobile devices (iOS, Android) to support touch-based gestures for the interactive interface. Setting up the connection between the different devices is made available using the DIAL protocol.[3]

The front-end is build on top of an existing cluster-based platform called Springfield[4], which links together a variety of RESTful web services where audiovisual content is stored automatically, processed and made accessible for streaming. The audiovisual content is segmented into both temporal and spatial fragments[2] and linked to annotations, all generated by the LinkedTV platform, which enables versatile enrichment on different dimensions of a video.

For the multiscreen toolkit, additional extensions are written for Springfield in Java. They will not only handle the sharing, communication and joining of applications, but also provide a way to push adaptive designs to client in order to abstract the client side code.

MULTISCREEN TOOLKIT

We will describe briefly five core elements of the multiscreen toolkit and how these elements support developers in the creation of applications using it.

4.1 Unified application model

The aim of the multiscreen toolkit is to allow the easy implementation of multiscreen applications by developers. Therefore it's important to find a correct balance between using widely known programming languages and commonly used tools most developers will be familiar with and to remove the difficulties in implementation that may arise with the different type of applications that will be developed.

In the case of multiscreen applications challenges like for example multiple users and multiple screens per application, synchronization of video and metadata and multiple device types needs to be handled. The toolkit focuses on these aspects and one of the major features is that applications, identified by a unique uri, once created, run on the server even when no clients are attached. This means every user joins the application instead of starting a new copy of the application. All users will watch the same application while user and screen objects get added to the application. This allows developers to simply create a community of people that only need to join the application without having to deal with the communication and synchronization in between.

The connection between the application and multiple users and screens is maintained by using a comet/long polling connections so all screens and users are updated live. This allows for features like object and screen sharing in interesting ways (see figure 2 for an example application).

4.2 Hardware abstraction

When developing multiscreen applications for HbbTV, mobile devices, remote-control or gesture interfaces the pitfall is that the software can get overly complex due to the support for different versions needed for the range of platforms, input methods and displays. The aim is to provide hardware abstraction as part of the toolkit for several well-known hardware components so different devices can be used without any need for code changes. A good example is the remote control.

[3] http://www.dial-multiscreen.org/dial-protocol-specification/DIAL-2ndScreenProtocol-1.6.4.pdf?attredirects=0&d=1

[4] The platform is developed by Noterik and will be available in open source in the future.

Figure 2: Sample application that shows an image shared over multiple screens.

The toolkit provides a library that can handle signals from old fashioned remotes (e.g. HbbTV, see figure 3), a simulated HTML5 remote and gesture types of remotes (e.g. Leap Motion[5]). The library provides basic functionality like play, pause and next but also supports more complex actions that can be implemented in each of the applications. This allows for a flexible demo setups where it's possible to use real hardware when available but not demanding it. This combined with the sole software demand of HTML5 means that prototypes made can easily be shared with possible clients or other members of the development team who don't always have the target hardware available at their location.

4.3 Message and notification system

Each component inside an application can be uniquely identified by a REST based uri. The root uri is defined by the application that gets an id based on the domain and user who initiated the application. Other users can join the application simply by opening the same uri in a HTML5 compatible browser. As new components (e.g. users or screens) join the application they are assigned their own identifier within the application name space. The idea behind addressing all the components, including GUI elements, is that this allows communication where messages and notifications can be addressed based on the identifiers.

Figure 3: Example traffic light application with remote control support.

[5] https://www.leapmotion.com

The toolkit provides a RESTful based message API to send messages to any object or a group of objects in the namespace of the application. The use of the message API allows that any component can send messages and receive notifications from any other component within the namespace. Mostly this will be used to send messages to components within the same application but it can also be used to communicate to other applications or even external systems. Messages to a component that is shown on multiple screens will all receive the message unless you specifically address a screen by its id we also allow for wildcards in different forms so its easy to talk to multiple targets in flexible ways.

4.4 Client-server code injection

Another goal of the multiscreen toolkit is to help developers to focus on code that adds server side functionality. The server side uses Java in a J2EE/Restlet concept this is easily done by creating a structure where all the heavy lifting is done by the toolkit. The toolkit follows the path of servlets, restlets and applets by extending a base class called HTML5Application. The result is that a simple 'HelloWorld' example can be reduced to around 10 to 20 lines even if it has full multiscreen and multi user support.

On the client side JavaScript doesn't have the full-range of options of Java so code injection between the server and client is used. For example to be able to do touch type events (tap, drag, swipe, pinch) normally extra event code is needed even with support packages like hammer[6] that is also used in the toolkit. You also need to defeat all the normal browser type events to give your application the correct 'app' feeling people want when using a smart phone or tablet. By using injection for the hammer code the client application code is reduced by up to 30%. The same is done for all other mandatory code (e.g. using jQuery[7]) needed in modern HTML5 style interfaces depending on device, application needs and debugging mode.

4.5 Support services

Since the multiscreen toolkit runs applications on the server it can easily be extended with extra services. Three example areas that are particular interested are 1) support for external user and community services, profile maintenance and joining/leaving groups and sharing. 2) Make the integration with external data sources easier. Many of the prototypes build with the multiscreen toolkit will revolve around LoD/metadata databases and having direct access using the communication layer outlined in section 6.3 provides less and cleaner code. 3) Normally multi screen applications can be difficult to test and debug due to their distributed nature, the multi screen toolkit offers mechanisms to make this more convenient for the developer. Using code injection the toolkit inserts try/catch error and tracking handlers that allow the possibility to view what is displayed on each of the connected screens and to change properties of the application on the fly that directly affect all connected screens. All errors are forwarded to the server and stored in a central place for easy discovery of errors on certain screens. This extends to normal admin tools (see figure 4) where the maintainer of the Springfield cluster can see what applications are running which users and screens are attached and can interact with them as part of the debugging process.

[6] http://eightmedia.github.io/hammer.js/
[7] http://jquery.com

Figure 4: The multiscreen toolkit admin console.

5. FUTURE WORK

The 2nd screen application demo will be released in June 2013, and is intended at this phase mainly for the LinkedTV consortium partners, who can use it in multiscreen application design within the project. The application and the multiscreen toolkit serve as a basis for the LinkedTV application and interface development during the last project year. We also hope to develop the toolkit further so that it can be easily deployed for quick prototyping of multiscreen concept and application ideas more widely.

Further research will be done in the areas of how we can use parts of the toolkit in the upcoming HbbTV 2.0 framework for both single and multiscreen solutions, and how the DIAL protocol[8] can be used when viewers are watching together in multiple locations to share/send apps. User trials will also be conducted in the LinkedTV project, which provide further feedback on the multiscreen interface design.

6. ACKNOWLEDGMENTS

LinkedTV is funded by the European Commission through the 7th Framework Programme (FP7-287911).

7. REFERENCES

[1] Baltussen, L., Leyssen, M., Ossenbruggen, J., Oomen, J., Blom, J., Leeuwen, P. van and Hardman, L.. 2012. Antiques Interactive. *In Proceedings of EuroITV 2012 demo session.* Available at: http://www.euroitv2012.org/AdjProc_EuroITV2012.pdf p23-p24

[2] Troncy, R., Mannens, E., Pfeiffer, S., and Deursen, D. van. 2012. *Media Fragments URI 1.0 (basic). W3C Recommendation.* Available at: http://www.w3.org/TR/media-frags/.

[8] http://www.dial-multiscreen.org