Creating web-based interactive public display applications with the PuReWidgets toolkit

Jorge C. S. Cardoso[1,2]
[1]CITAR - Universidade Catolica Portuguesa
Rua Diogo Botelho, 1327
4169-005 Porto, Portugal
jorgecardoso@ieee.org

Rui Jose
[2]Centro Algoritmi
Campus de Azurem
4800-058 Guimaraes, Portugal
rui@dsi.uminho.pt

ABSTRACT

Interaction is repeatedly pointed out as a key enabling element towards more engaging and valuable public displays. Still, most digital public displays today do not support any interactive features. We believe that this is mainly due to the lack of efficient and clear abstractions that developers can use to incorporate interactivity into their applications. In this demo we present PuReWidgets, a toolkit that developers can use in their public display applications to support the interaction process across multiple display systems, without considering the specifics of what interaction modality will be used on each particular display. PuReWidgets provides high-level widgets to application programmers, and allows users to interact via various interaction mechanisms, such as graphical user interfaces for mobile devices, QR codes, SMS, etc.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Software libraries, Modules and interfaces*

General Terms

Human Factors, Design

Keywords

interactive public displays, mobile applications, toolkits

1. INTRODUCTION

Although public digital displays have become increasingly ubiquitous artefacts, most of the displays we encounter today still do not offer any interactive feature. A key reason behind this apparent paradox is the lack of efficient and clear abstractions for incorporating interactivity into public display applications. While interaction can be achieved for a specific display system with a particular interaction modality, the lack of proper interaction abstractions means that there is too much specific work that needs to be done outside the core application functionality to support even basic forms of interaction. This is an effort that must be replicated by each developer, representing a wasted effort. This also leads to inconsistent interaction models across different displays and, as a result, people are not able to develop, based on previous experiences, any expectations and practices regarding their interaction with public displays.

In our work [2, 3], we have studied what kinds of abstractions would be useful for public displays and we have developed a toolkit which incorporates those abstractions. PuReWidgets allows application developers to focus on the core functionality and input data needs of the application, giving them high-level abstractions (widgets) that hide the details of the multitude of interaction mechanisms that can be used to interact with a public display. We believe this to be an important step in the vision of open networks of interactive public displays [4].

2. PuReWidgets

PuReWidgets is a widget toolkit for the development of web-based interactive public display applications. A widget is an interaction abstraction that: provides applications with high-level interaction data, is independent of the underlying input mechanism; can have different graphical representations in different platforms (including the public display itself); and provides system-level input feedback to users.

The development process of a public display application that uses PuReWidgets is similar to the development of a regular web application[1]: the developer includes the PuReWidgets external code library in his project and uses the available functions of the library to code the application, instantiating widgets and registering interaction event callbacks. The developer then deploys the set of HTML, CSS, and Javascript files on a web server. The PuReWidgets system is composed of a widget library that programmers include in their application's code, and a web service that handles interaction events (Figure 1). When the application is running, the PuReWidgets library receives input events from the PuReWidgets service, and triggers the appropriate high-level event on the application.

MUM '12, Dec 03-06 2012, Ulm, Germany
ACM 978-1-4503-1815-0/12/12.

[1]PuReWidgets is implemented for Google's Web Toolkit (`http://code.google.com/webtoolkit`) and Google's Appengine (`http://code.google.com/appengine`) web development frameworks.

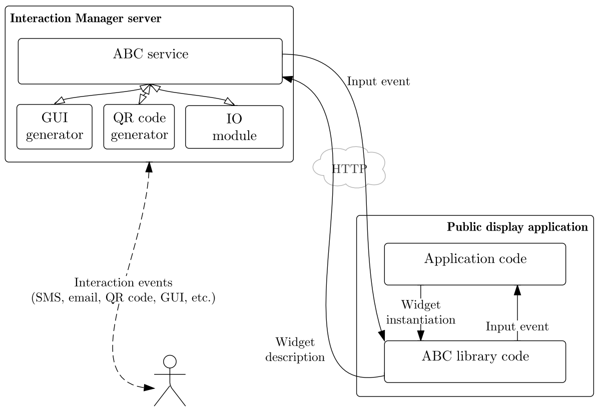
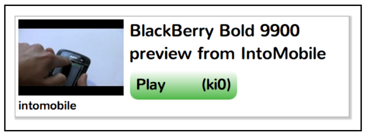


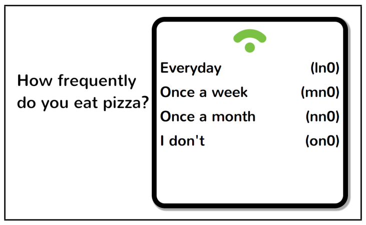
Figure 1: PuReWidgets architecture.

2.1 Widgets

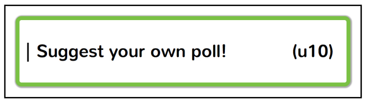
PuReWidgets provides widgets that support the most common input types for public displays. Widgets have a default graphical representation that can be shown on the public display, but it can be overridden by applications to provide a better integration their own "look and feel". Figure 2 shows a sample of widgets that have been restyled in our demo applications (Public YouTube Player and Everybody Votes applications, described later).



(a) Button, in the context of a video item in the Public YouTube Player application



(b) Listbox, in the Everybody Votes application



(c) Textbox, in the Everybody Votes application

Figure 2: Graphical representations for widgets used in the sample applications.

The currently available widgets are:

Action button A button widget allows applications to provide actions that users can trigger at any moment. The button widget generates a simple trigger event identifying the action to be executed. Figure 2a show an example of a button.

Listbox A list widget allows applications to provide a list of items that users can choose from. The list widget triggers an event identifying the option selected by the user. Figure 2b shows an example of a listbox (in this particular case, the listbox was styled to show the title on the side, instead of the default on top position).

Textbox A textbox widget allows applications to receive text from users. A textbox widget generates an event containing the input text. Figure 2c shows an example of a textbox.

Upload An upload widget allows applications to receive media files. The input event carries the URL of the media file.

Download A download widget allows applications to specify external locations where content items can be accessed and downloaded. Applications specify the location of the file when instantiating the widget. The download widget triggers an event with a simple indication that a user has asked for the file.

Checkin A checkin widget allows applications to receive notifications when users check-in in the place where the display is. In this case, the input event the application receives carries the identification of the user that has just checked-in.

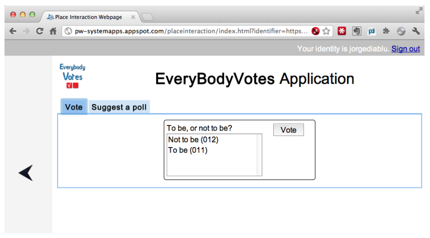
2.2 Interaction mechanisms

Users can interact with the widgets created by an application using a variety of input mechanisms: using a graphical user interfaces for mobile (and desktop) devices, scanning a QR code associated with a particular widget, or sending a specially formatted text message via SMS or email. (We have focused on remote interaction mechanisms, but the widgets can also be used with a touch-screen, or standard mouse and keyboard platform.)

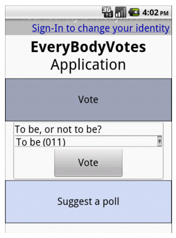
2.2.1 Generated GUI

The Interaction Manager (IM) server (see Figure 1) is capable of generating a web-based graphical user interface for every application that uses PuReWidgets. Figure 3 shows an example of the interface generated for the Everybody Votes application in both a desktop and a mobile device.

These interfaces are generated by inspecting the widget descriptions stored at the IM, associated with a particular application. These descriptions are automatically sent to the IM by the PuReWidgets library when a widget is instantiated.


Place Interaction Webpage
Your identity is jorgediablu Sign out
EveryBodyVotes Application
Vote
Suggest a poll
To be, or not to be?
Not to be (012)
To be (011)
Vote

(a) Desktop



(b) Mobile

Figure 3: Generated GUI interfaces.

2.2.2 *QR Codes*

The IM also provides display owners with the possibility of obtaining QR codes for any widget that has been created by any application. This allows, for example, the creation of flyers or posters for particular features of an application that can then be distributed near the display. These flyers or posters can be used to raise awareness about an application or a particular feature, and entice users to interact. Figure 4 shows an example of a flyer created for a poll about the winner of a national football championship.

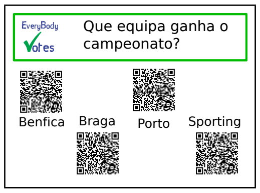

EveryBody Votes
Que equipa ganha o campeonato?
Benfica
Braga
Porto
Sporting

Figure 4: Sample flyer with QR codes for a poll. The QR codes are generated automatically by the PuReWidgets toolkit.

2.2.3 *Text-based mechanisms*

An I/O module in the IM handles text messages from SMS and email (which can also contain attachment files), and parses them using a predefined command syntax that allows users to address specific widgets in an application.

For SMS messages, uses can address a specific widget with a <place_id>.<reference> <parameters> message, where <place_id> is a string defined by the display owner that uniquely identifies that place in the PuReWidgets system, <reference> is a 3-character code that is shown inside the widget in the public display, and <parameters> is an optional list of space separated strings which are interpreted differently by different widgets. For example, in Figure 2, to select the "Everyday" option of the listbox for a display in the place identified by "UCP", users would write "ucp.ti0"; to send a poll suggestion, users would write "ucp.u10 suggestion text.". The SMS message would then be sent to a phone number announced near the display.

Email messages follow a similar mechanism. PuReWidgets assigns a unique email addresses to each place, and parses the subjects and attachments of emails sent to that address. The subject follows the format <reference> <parameters> (there is no need to specify the place id because that is implicit in the email address). In case the widget accepts files (e.g. an upload widget), the attachments are extracted and stored and their URLs are made available to the widget.

Other mechanisms such as Bluetooth naming and Instant Messaging could easily be integrated as an I/O module in the Interaction Manager server.

2.3 Asynchronous interaction

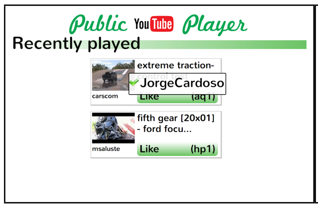
PuReWidgets' widgets are capable of receiving input even if the application is off-screen (i.e., not being rendered on the public display), however they do no generate events or graphical feedback in this situation. The IM maintains an input queue for every widget. When the application is put on-screen, the PuReWidgets library requests all input that was sent while the application was off-screen, and delivers it to the application.

2.4 Input feedback

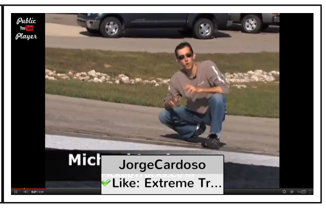
An important aspect of desktop widgets is the system-level feedback they provide and that helps users understand the response of the system, independently of how the application will react. Our approach is to provide a base mechanism for presenting feedback on the public display, that consists of a panel that pops up on top of the corresponding widget. This feedback allows not only the interacting users to get a confirmation of their actions, but also non-interacting users to get a better understanding of what is going on the public display. The information displayed in the panel is defined by each widget but, generally, it identifies the user (a masked version of the user id if the identification is considered sensible information, such as the phone number for SMS interactions) and the action that the user performed. The toolkit provides slightly different feedback information on the public display, depending on whether the widget that is receiving the input is currently visible on-screen, or if it is not. For on-screen widgets, the widget itself provides context to the feedback information, so the feedback panel contains, generally, less information (Figure 5a). For off-screen widgets, the feedback panel must provide information that helps users identify the widget itself (Figure 5b).

The feedback also helps users understand the asynchronous nature of the interaction by providing time information when the input which is causing the feedback on the display is older than a configurable amount of time (1 minute by default). In this case the feedback includes the age of the feedback, e.g., "about 10 minutes ago...".

The toolkit provides default feedback messages for all widgets, but programmers can customize those messages with their own text.



(a) Feedback for on-screen widgets



(b) Feedback for off-screen widgets

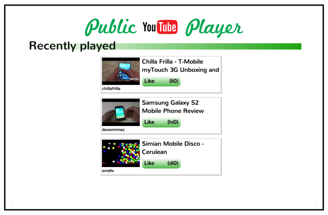
Figure 5: Input feedback on the public display.

3. DEMO SETUP

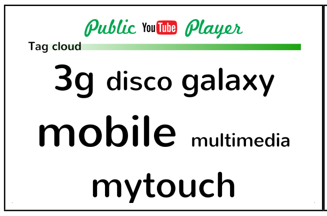
In this demo, we will setup a public display running two interactive applications (Public YouTube Player, and the Everybody Votes applications) that can be interacted with using SMS, Email, QR codes, or a web interface. The accompanying video [1] shows some of the interactions that will be possible during the demo.

3.1 Public YouTube Player

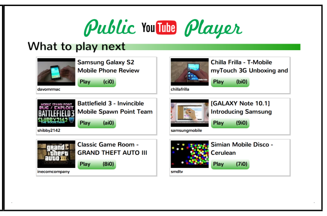
The public video player is an application that searches for, and plays youtube videos. The application plays videos from a set of youtube users specified by the place owner, and videos that result from searches based on tags. The search tags are taken from a tag cloud that is built using tags defined by the display owner, and tags extracted from the previously played videos.


Public YouTube Player
Recently played
Chilla Frilla - T-Mobile myTouch 3G Unboxing and
Samsung Galaxy S2 Mobile Phone Review
Simian Mobile Disco - Cerulean
Like

(a) Recently played videos



(b) Tag cloud


Public YouTube Player
What to play next
Samsung Galaxy S2 Mobile Phone Review
Chilla Frilla - T-Mobile myTouch 3G Unboxing and
Battlefield 3 - Invincible Mobile Spawn Point Team
[GALAXY Note 10.1] Introducing Samsung
Classic Game Room - GRAND THEFT AUTO III
Simian Mobile Disco - Cerulean
Play

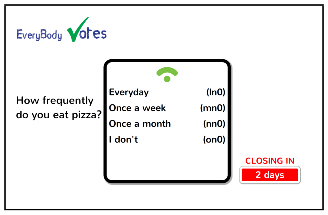
(c) Videos to play next

Figure 6: Screens in the Public YouTube Player application.

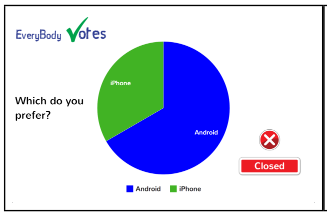
The application is composed of four screens that iterate over time: (6a), a screen that shows the three most recently played videos, and acts as an activity stream for the applications; (6b), a screen that shows the current tag cloud, giving users an indication of the topics that have been played; (6c), a screen which shows a set of alternative videos that can be played next; and finally, the screen which plays the selected video in fullscreen.

Users can interact with the application at any time (including when the application is off-screen) to: "like" a video that has already been played; get the URL of a recently played video;select a video to play from the list of search results;report an innapropriate video.

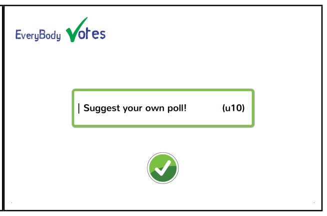
3.2 Everybody Votes



(a) Open poll



(b) Closed poll



(c) Suggestion box

Figure 7: Screens in the Everybody Votes application.

The Everybody Votes application is a polls application composed of three screens, depicted in Figure 7: (7a), an open polls screen which iterates through the open polls, showing the poll question, possible answers, and time left before the poll closes; (7b), a closed polls screen which iterates through the closed polls and shows their voting results; (7c), a suggest box screen enticing users to suggest their own polls (which will go through a moderation process by the display owner).

As with the Public YouTube Player, users can interact with this application at any time to vote on the various open polls, or to suggest a poll to the display owner. Display owners receive the poll suggestions in an email address configured in the application's administration interface.

4. CONCLUSIONS

Interactive public display applications need to be easier to develop and need to start providing consistent interaction models to its users. For this we need programming tools anchored on the study of interaction with public displays, that programmers can incorporate in their applications. The PuReWidgets demo shows an initial version of such a toolkit, demonstrating its features and applicability.

5. ACKNOWLEDGMENTS

Jorge Cardoso has been supported by "Fundacao para a Ciencia e Tecnologia" (FCT) and "Programa Operacional Ciencia e Inovacao 2010" (POCI 2010), co-funded by the Portuguese Government and European Union by FEDER Program and by FCT training grant SFRH/BD/47354/2008. This research has also received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 244011 (PD-Net).

6. REFERENCES

[1] J. C. S. Cardoso. PuReWidgets toolkit demo video for MUM2012. doi:10.6084/m9.figshare.94598, 2012.

[2] J. C. S. Cardoso and R. Jose. A Framework for Context-Aware Adaptation in Public Displays. In R. Meersman, P. Herrero, and T. Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, volume 5872/2009 of *Lecture Notes in Computer Science*, pages 118–127, Vilamoura, Portugal, 2009. Springer Berlin / Heidelberg.

[3] J. C. S. Cardoso and R. José. PuReWidgets: a programming toolkit for interactive public display applications. In S. R. José Creissac Campos, Simone D. J. Barbosa, Philippe Palanque, Rick Kazman, Michael Harrison, editor, *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems - EICS '12*, page 51, New York, New York, USA, June 2012. ACM Press.

[4] N. Davies, M. Langheinrich, R. Jose, and A. Schmidt. Open Display Networks: A Communications Medium for the 21st Century. *Computer*, Mar. 2012.