The PuReWidgets Toolkit for Interactive Public Display Applications

Jorge C. S. Cardoso
CITAR – Portuguese Catholic University,
Rua Diogo Botelho 1327 Porto, Portugal
jorgecardoso@ieee.org

Rui José
Algoritmi – University of Minho,
Campus de Azurém, 4800-058 Guimarães, Portugal
rui@dsi.uminho.pt

ABSTRACT
In this paper we present PuReWidgets, a toolkit that supports multiple interaction mechanisms, asynchronous events, automatic web and mobile interface generation, and concurrent interaction. This is an early effort towards the creation of a programming toolkit that developers can incorporate into 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.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Software libraries, Modules and interfaces*;

General Terms
Human Factors

Keywords
Human-Computer Interfaces, User Interface Design, Programming Toolkits, Public Displays

1. INTRODUCTION
Public digital displays have become increasingly ubiquitous artefacts in public and semi-public spaces. Most of them, however, do not support any interactive features, even though interaction is clearly recognised as a key element in making them more engaging and valuable. A key reason behind this apparent paradox is the lack of efficient and clear abstractions for incorporating interactivity into public display applications. In this work, we studied new interaction abstractions for the development of interactive applications for public displays. Our early results are instantiated in a programming toolkit that provides a widget-like high-level interaction abstraction and that developers can incorporate into their public display applications.

Our target public display environment is an environment that is open to place owners, to application (or more generally, content) providers, and to users. To facilitate application development and deployment, we assume that public display applications are web-based and can be hosted on third-party servers to serve content to many displays, taking advantage of the locally available interaction resources. Software developers will create these applications and will want to be able to distribute them globally. Place owners will be able to browse, select, and configure the applications they want to display in a given location. An application selected for a place will be sometimes visible on a public display. We assume that each display will show content from multiple applications and will iterate through those applications based on some pre-defined scheduling criteria. Even though an application may not be continually visible on the public display, it will be accessible via many other displays and interaction mechanisms. Once selected for a place, the public display application will be able to receive and process interaction events and produce place specific content that can be accessed in different ways (on a public display, through a web page, through a custom mobile application, etc.).

ISPD'12, June 4-5, 2012, Porto, Portugal

2. RELATED WORK
Interactive public displays are not new, and there are many systems that explore different interaction mechanisms that can be used by applications for public displays. For example, Rohs [1] has implemented a set of widgets for visual marker-based interaction that allows users to activate actions or select options encoded in a visual marker and send it via SMS (using a custom mobile application). Dearman & Truong [2] developed Bluetone: a widget that is activated through dual tone multi-frequency (DTMF) over Bluetooth. Users interact with an application by changing the Bluetooth name of their device to a system command, wait for the display to pair with the user's phone as an audio gateway, and then pressing the keys on the keypad of their phone. SMS interaction has also been used frequently with public display applications. Bluetooth naming is another approach for providing interactivity to public displays. Lancaster University's e-Campus display system [3], for example, explored Bluetooth naming as an explicit input mechanism. All these are good examples of how to provide users with specific interaction channels to public displays. However they do not address the question of providing useful high-level interaction abstractions to applications, independent of the interaction mechanism.

3. PUREWIDGETS SYSTEM
The PuReWidgets system is composed of a widget library and web service that handles interaction events. A widget is an interaction abstraction that: provides developers with high-level interaction data, hides the specific details of the underlying input mechanism; and can have different graphical representations in different platforms (including the public display itself). The development process of a public display application that uses PuReWidgets is similar to the development of a regular web application: the developer includes an 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 life cycle of a public display application (start, stop, and reacting to input events), however, is very different from the life cycle of a desktop application: the application is instantiated and terminated by a scheduler software that drives all the content of the public display, and interaction events can be generated via multiple local or remote sensors. When a widget is instantiated by an application, its description data is sent to the PuReWidgets service, which keeps track of every widget instantiated by every application. An I/O infrastructure is responsible for accepting raw input events from

users and relay them to the PuReWidgets service, which routes them to the application/widget that was addressed by the user. This service acts as an input event queue, storing the widget input until the application is ready to receive them. This allows applications to receive widget events even they were generated when the application was not executing at the public display. When the PuReWidgets library asks for input, the service replies with the stored input. The library (running within the application) then forwards the input to the correct widget instance so that it can trigger the high-level application event. This requires a distributed architecture in which some widget information is kept by remote services, effectively decoupling widgets from applications. This allows the toolkit automatically generate of various interaction points (web, mobile, etc.) for an application. The necessary components for this system are illustrated in Figure 1.

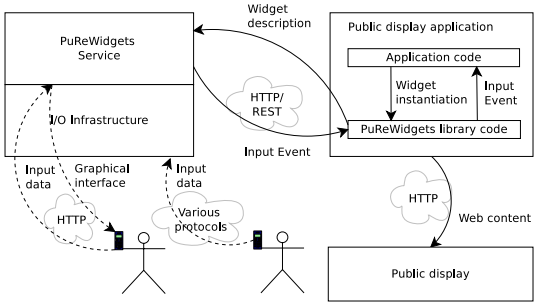

Figure 1. Global architecture of the PuReWidgets toolkit

3.1 Toolkit Features

The PuReWidgets toolkit provides a number of important features for interactive public display applications.

3.1.1 Control types

Widgets are provided in the form of an object-oriented library in which each widget has a type that defines the type of high-level data that it exposes to the application. Programmers can choose which widgets to use, according to the application's data needs (in some case there may be alternative widgets for the same data need), by instantiating the respective widget class and registering a call-back to receive the high-level events generated by the widget instance. The toolkit provides five main control types: Imperative/Selection controls allow users to select among several related options or issue commands to the application; Entry controls allow users to input simple data such as free text or bounded values; Upload controls allows users to submit media files to the public display application; Download controls allow the application to provide files that users can download to their personal devices, or forward to their email, etc.; Check-in controls allow users to signal the application that they are present. Application developers can create other types.

3.1.2 Multiple interaction points

PuReWidgets stores widget metadata on a server, allowing the display system infrastructure to keep track of the widgets that each application is using and providing alternative interaction points beyond the public display itself. PuReWidgets provides automatically generated application interfaces for desktop and mobile devices. It also automatically generates QR codes for direct interaction with widgets.

3.1.3 Addressing an input routing

PuReWidgets provides a simple addressing mechanism that can be used with input mechanisms such as SMS, Bluetooth naming, email, etc. The mechanism is based on textual references generated automatically for each widget. The toolkit also takes care of routing the input to the respective application/widget.

3.1.4 Asynchronous interaction

PuReWidgets can receive input even when the application is not displaying any content on the public display. Applications can react through other means (such as webpages), or wait to be displayed.

3.1.5 Graphical input feedback on the public display

On the public display, widgets can be configured to automatically display interaction feedback at the system level, i.e., regardless of the specific reaction of the application. Several parameters for this feedback can be set, but by default, it displays a graphical popup panel near the graphical representation of the widget that has received input. This is an important feature not only because it gives confirmation that the input was received (which some input channels may not support, e.g., Bluetooth naming) but also because it provides interaction awareness to other people near the display.

3.1.6 Extensibility

The toolkit allows programmers to extend the available widgets creating new behaviors and graphical representations for the public display, desktop and mobile device interaction.

4. CONCLUSION

We have created a toolkit for developing interactive public display applications, which handles much of the work a developer would have to deal with to develop even the simplest interactive public display application. PuReWidgets provides high-level interaction abstractions that suit the kind of interaction one normally does with public display applications and transparently supports various interaction mechanisms. The toolkit provides a widget addressing and an input routing mechanism, supports concurrent, asynchronous interaction and provides decoupled graphical affordances that can be used directly on the public display, or on alternative platforms. This toolkit fills a clear gap in the area of interactive public displays. Having a foundational tool like PuReWidgets allows designers and programmers to focus on the real creative work of designing interesting applications and user experiences.

5. Acknowledgements

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

6. REFERENCES

1. Rohs, M. (2005). Visual Code Widgets for Marker-Based Interaction. 25th IEEE International Conference on Distributed Computing Systems Workshops (pp. 506-513). Washington, DC, USA: IEEE.
2. Dearman, D., & Truong, K. N. (2009). BlueTone. Proceedings of the 11th international conference on Ubiquitous computing - Ubicomp '09 (p. 97). New York, New York, USA: ACM Press.
3. Storz, O., Friday, A., & Davies, N. (2006). Supporting content scheduling on situated public displays. Computers & Graphics, 30(5), 681-691.