Selection and control of applications in pervasive displays

Constantin Taivan[1], Rui José[1] and Ivan Elhart[2]

[1] Centro Algoritmi, Campus Azurém, Guimarães, Portugal
{constantin,rui}@dsi.uminho.pt
[2] University of Lugano, Via G. Buffi 13, 6904, Lugano, Switzerland
ivan.elhart@usi.ch

Abstract. Public displays are progressively embedded in urban settings. Such displays become elements of an integrated pervasive ecosystem in which various displays with multiple applications are accessed by multiple viewers. Still, many public displays employ content that is based on pre-defined schedules as encountered in conventional digital signage systems. We envision future display deployments embedding many applications that are running concurrently and able to continuously react to users' requests. In this paper, we investigate application selection and control concepts based on a mixed-initiative scenario in which display system and viewers are both involved in the process of content presentation. Our approach is inspired by traditional GUI interaction concepts and design considerations of sensing systems. Hence, this research would inform the design of novel techniques for application selection and control in pervasive display environments.

Keywords: public displays, ubiquitous computing, mixed-initiative interaction.

1 Introduction

Urban spaces are increasingly embedded with ubiquitous computing technologies and in particular with various types of public digital displays. This is leading to the emergence of pervasive display systems that can be described as perch/chain sized ecosystems for many-many interaction, composed of displays of various sizes (from handheld devices, to medium/large wall mounted displays), and where "many people can interact with the same public screens simultaneously" [1]. Displays in pervasive display systems are inherently multi-purpose and our vision is to move away from a world of closed display networks to a world of open display networks in which large-scale networks of pervasive public displays and associated sensors are open to applications and content from many sources [2].

In this type of environment, content shown on public displays should not correspond to pre-defined schedules, as is the case of today's conventional digital signage systems. Instead, there will be many applications running concurrently and being able to react at any moment to input from users. The display environment may thus be seen as a mixed-initiative scenario [3] in which the system and its associated applications are always showing content that is not only of a high relevance for the current

display settings, but also content that reflect the current user interactions in that environment.

Consequently, a pervasive display environment should manage the temporal and spatial allocation of the displays between applications, as well as support selection and control techniques that would allow people to drive the way applications are shown and used in that environment. This should all be done considering the need to accept, acknowledge, and resolve concurrent input from multiple users in a way that is fair and clear for all those users.

In this study, we aim to inform the definition of novel techniques for application selection and control in pervasive display environments that can address the above challenges. These should enable multiple users to concurrently drive the selection of the applications being shown and control of their behavior. We avoid as much as possible assumptions about particular application models. For example, we are not considering whether applications are running on the cloud, on the displays themselves or any combination in between. Similarly, we are not considering how different systems may address the implications of multi-user concurrent interaction, but only that such concurrent interaction will exist and will expose challenges approached in our study.

Pervasive display environments may be seen as a mixture between traditional GUI systems and sensing systems. Therefore we chose a methodology that is based on the study of well-known selection and control concepts from GUI systems and on the analysis of their applicability to the specific characteristics of pervasive displays. After presenting related work in Section 2, we present in Section 3 a study of concepts in traditional GUI systems. In Section 4, we analyze the case of GUI concepts for public displays. The paper ends with conclusions in Section 5.

2 Related Work

Researchers in the area of public displays have largely investigated how to appropriate displays by creating rich interactive applications and services [4][5][6]. Displays as multi-user and shared resources have also been explored mainly from two perspectives: time based queuing [7] and explicit [8] or implicit space partitioning [9]. Morales-Aranda et al. [10] describe a display prototype as an implicit space partitioning system that can dynamically adapt content layouts allowing two users to visualize their personal information. Vogel et al. [9] provides insights on how a public ambient displays can be either individually or collaboratively shared. This functionality reflects the same assumptions that we are considering in our research, i.e., the display is a shared resource and is not solely appropriated by one user at a time.

The study by Peltonen describes CityWall [11], a large multi-touch display installed in a central location in Helsinki, Finland. The system can visualize images retrieved from Flickr and the users can interact with the content using one- or two-handed gestures, e.g., resize, rotate, etc. This research is relevant for us because it discusses how people succeed to appropriate a large display, resolve the potential conflicts, and find the right moment to take the turn. Their results show that the decision about when a person should interact does not depend only on the available space

at the display, but rather on a set of complex social interactions of reasoning and negotiation between participants.

Sacks et al. [12] investigate a turn taking system for casual conversation seeking an answer on how do participants in the conversation "select" the next speaker? They build a set of rules that could help organizing the speaker selection. This work is an analogy to ours and we acknowledge its relevance to convey place specific interactions. To the best of our knowledge, the design of generic techniques for application selection and control in multi-user, multi-application scenarios has not been addressed.

3 Selection and Control in GUI Systems

The methodology is based on the identification of concepts of application control in traditional GUI systems and the analysis of their applicability to the domain of public displays. As the first phase of our study, we selected descriptions of techniques for application selection and control in desktop and mobile interaction GUIs. These were retrieved from various sources, including the Wikipedia, books, and various web sites. We used a total of 31 descriptions referring to 20 different concepts[1] from various interaction models and operating systems (OS), e.g. Windows, UNIX, Mac OS, iOS and Android.

In the second phase, we analyzed each description to identify the multiple ways in which applications can be selected and controlled. Each reference to a selection or control technique was coded using open coding and a dedicated coding software. For every code created, a small memo describing its generic meaning was associated. This resulted in 61 codes. At the end, we conducted several consolidation sessions to establish the main concept clusters in relation to application selection and control, resulting in 5 main categories: *Controlling Application Life Cycle*, *Application Identification*, *Implicit Application Selection*, *Explicit Application Selection* and *Visual Layers*. These categories will be used in the next section as a frame for the analysis of the specific challenges involved in application selection and control in pervasive displays.

Controlling Application Life Cycle. The application life cycle embodies the sequence of events that occur between the launch and termination times of an application. During the application life cycle, an application may be executed in different states, e.g., background execution, suspended, inactive, foreground execution. An important group of techniques is primarily aimed at enabling control of the transitions between the various phases of the application life cycle.

For instance, in iOS the application life cycle is composed of five distinct states: *not running* – the state of a rebooted device, *active* – the application is displayed on the screen and receive inputs, *background* – the application may execute code without receiving inputs or update the screen, *suspended* – an application is frozen and its

[1] The GUI concepts are published at:
http://ubicomp.algoritmi.uminho.pt/research/apps4publicdisplays/, 26.06.2012

state is stored in RAM and *inactive* – a temporary rest between two other states, e.g., yielded by incoming calls or if the user has locked the screen. While in the active state, an application may require visual and input resources, in the background execution the application is running in a constrained behavior without requiring any display or user input resources. In a UNIX environment, a process, i.e., a program in execution may run behind the scenes fulfilling various activities such as logging, system monitoring, scheduling and user notification. Supporting multitasking, i.e., the ability of an OS to execute multiple applications at once, is concerned not only with the operating system on which the applications are installed but as well with the hardware platform. For instance, background code execution is limited to a certain types of devices, e.g., iPod third generation and iPhone 4. As well, an application can go to the background mode as users switch between applications.

Application Identification. Application identification is concerned with ways in which users can recognize and distinguish the various apps. Normally, the applications from traditional computing platforms may be identified through icons, a thumbnail photo briefly describing its functionality, or during execution by using specific system level indicators of common app description fields, e.g., windows title, favicon.

For instance, in Windows and Mac desktop environments there is a system-based software application that lets users identify and inspect all applications in execution and their respective processes or tasks. Application icons allow users to easy recognize and launch applications. It is represented as a small picture, which intuitively describes the function of the respective program. An application icon is designed to be language independent (does not contain any text) and it offers rapid entries in the system functionalities. Application icons may depend on the application execution state. For example, in Windows desktop environments an application is invoked through a static icon and can be monitored through a dynamic or state icon, e.g., often used for background execution. In particular, Windows 8 features a new user interface paradigm based on the concepts from Windows Mobile Phone – a design based on live tiles. A live tile is a dynamic icon with a larger size that identifies the respective app and shows app specific data in the same time, e.g., a live tile showing the number of unread email messages.

Implicit Application Selection. Selection of applications can be done on the basis of system or application centric initiatives. The system-based activation is an additional way to launch applications as a result of various event triggers. This may include time-based events or certain system specific service events such as Device Join, Firewall Event, Network IP Availability, etc. The application based selection entails application specific logic that triggers its appearance, e.g., from background to foreground.

When an event occurs it may trigger background or foreground application activation. In Windows OS as a consequence of system-based scheduling, most processes are launched in background mode without any user interface, e.g. Server, Network

Connections. Foreground processes designate applications that features visual user interfaces and require direct user interactions.

Explicit Application Selection. Application selection is an action in which a certain application changes its execution state and visual appearance, i.e., from background to foreground. There are two common possibilities in launching the applications: 1) from the whole list, e.g., all applications installed on the system and 2) from a sub-set list, e.g., executing apps, recent list. The first approach includes GUI elements like *Start Button, Start Menu* whilst the second includes *app switch, app dock, folders, taskbar, etc*. The *application icon* element resides in both aforementioned cases.

For instance, in Windows desktop GUI, an application contains multiple places from where it can be launched, e.g., *Start Menu, hotkeys*. In Windows Vista the App Launcher gadget facilitates common task grouping and execution. Taskbar element serves the applications purposes differently in Windows and MAC. While in Windows the taskbar is window oriented, in MAC, the dock is application oriented, that means an application will not display all containing windows. Usually in Windows environments the taskbar contains a start button to access the start menu, quick launch panel, taskbar buttons and a notification area. In Windows OS, Mac, KDE, UNIX there is a keys combination, i.e., Alt+Tab, that offers an app selection alternative by switching between the most recent top-level application windows.

Visual Layers. The concept of visual layers describes all the application graphical appearance in the screen layout. It is common in desktop OSs to distinguish between applications by using the criteria of the most active application window, i.e., always-on-top. Although an application might have multiple windows, the user attention is focused only on the one that is in foreground. One application may feature additional visual layers such as splash screens, or a sort of preliminary windows in which users may optimize the application execution.

Particular types of visual layers are notifications. A notification is a typical application feature. In a traditional OS, a notification message warns users about application data updates or about system level issues. Mainly, the computer notifications contain two classes: a) one that calls for user attention, e.g., pop-ups and b) the other that does not call for explicit user attention, e.g., pop-under. A pop-under notification contains a non-intrusive content that resides behind scene. In Windows environments, non-intrusive notifications are shown in the notification area situated in the right side of the Taskbar.

4 Challenges for Pervasive Display Systems

This section analyses the main challenges involved in supporting different facets of application selection and control in the context of applications for pervasive display systems. The methodological approach is inspired by the work of Bellotti et al. on sensing systems framework [13]. We used the categories identified in the previous

section as the framework for the major challenges in application selection and control. For each of those categories, we review the traditional GUI solutions and analyze the new challenges raised by the specificities of public display applications.

When considering those specificities, we assume in particular that there can be many concurrently interacting users in the environment and also that the execution environment of the applications is not a single display, but instead an ecosystem of displays with multiple distributed user interfaces that span across multiple devices, e.g., public displays, mobile phone, touch enabled surfaces.

In regard to the control of the **application life cycle**, the main implication is a separation between the execution state in the environment and the execution state on any particular device. While applications may be expected to be always available and ready to produce content on any display, their normal execution mode may be a waiting mode in which they are ready to receive input signals and in appropriate moments generate content for presentation on the displays. The main challenges will be how to model this combination between execution states and presentation state, in a way that people can easily perceive and learn to control. It is also necessary to separate application availability in the environment from its presentation on the displays or from its execution on any particular device of that environment.

Identifying applications is important so that people may associate the content they see on the displays with the application generating that content. An adapted version of GUI concepts, such as application titles may be used in some cases, but may also be inappropriate in other cases because it may interfere with the rich visualization requirements of public displays. Alternative approaches may include a list of the applications that are currently available to be shown on the displays. This list may include the application id and a summary of its content, e.g. live tile, and may be available through mobile devices or occasionally shown on the display to prompt interaction.

Implicit application activation allows the presentation of a particular application on the displays to be triggered by an external event. In a mixed-initiative model, the system would need to implicitly call for specific apps, even if there is no activity from users. Additionally, some applications may only be relevant when particular contextual conditions occur. In such case the system may at any moment make selections based on the interpretation of the respective context, e.g. people present and their preferences. Therefore, a challenge for pervasive displays is the ability to integrate this dynamic application selection into the application execution mode.

Explicit application selection in public displays is based on viewers' explicit requests for applications. Firstly, viewers need to identify applications in order to be displayed. Afterwards, using various interaction techniques, e.g., mobile phone through a Bluetooth connection, gestures, they can call for a particular application to be shown. The main challenge is mediating between possible conflicting requests from multiple users or even between users and system goals.

Multiple visual layers are an important feature, mainly in desktop systems, because they allow a more sophisticated management of the interaction with people. However, in public displays, especially with multiple people sharing the display, it becomes much more challenging to achieve a balanced combination between multiple

layers and a good interaction experience. Still, well-designed notification layers that choose the best time to present themselves may provide an important alternative channel for presenting contextually relevant content outside the normal presentation cycles of the applications. In particular, these alternative visual layers may be important in generating feedback for users trying to interact with the system and support progressive interaction modes in which users and displays are increasingly aligned while minimizing interactions by accident such as in gesture-based interfaces.

In Table 1, we summarize these basic questions on application selection and control and the specific challenges they raise for public displays systems.

Table 1. GUI solutions and public display challenges for application selection and control

Basic Questions	**Traditional GUI Concepts**	**Challenges for public displays**
How can the system or the people using it control the application life cycle?	System priorities; Triggers; Execution as service Execution modes	How to model the relation between the content presentation and application current state?
How do viewers identify applications?	Static and dynamic icons, e.g., live tiles, windows title bar, favicon, etc.	How to raise awareness about installed or running applications? How to associate content being presented with the application? How to address an application?
How does system implicitly select applications?	System events, e.g., Device Join, Firewall events, Network IP Availability, etc.	How to model display ecosystem events? How to select relevant apps based on current context?
How do viewers explicitly select applications?	Start Button, Start Menu, App Switch (Alt+Tab), app dock, folders or taskbar.	How to mediate between possible conflicting requests of the system users?
How to use multiple layers to enhance application selection and control?	Applications windows and notifications; system level notifications, e.g., pop-ups, pop-under.	How to effectively use multiple visual layers without disturbing the current content presentation and overall user experience?

5 Conclusion

In this paper, we analyzed traditional GUI concepts for application selection and control and discuss how they could serve as the basis for addressing similar challenges in multi-application display environments. The results highlight that there are many similarities and therefore many common solutions that should be adapted for this new application domain, but also identify a number of unique challenges that may need more that simple adaptions. Further work will use these design considerations for implementation and validation of the envisioned application selection and control techniques.

Acknowledgments

This research has received funding from PD-NET project, which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 244011 and from "Fundação para a Ciência e a Tecnologia" under the research grant SFRH/BD/75868/2011.

References

1. Terrenghi, L., Quigley, A., Dix, A.: A taxonomy for and analysis of multi-person-display ecosystems. Journal of Personal and Ubiquitous Computing. 13, 583 - 598 (2009).
2. Davies, N., Langheinrich, M., Jose, R., Schmidt, A.: Open Display Networks: A Communications Medium for the 21st Century. IEEE Computer. 45, 58-64 (2012).
3. Horvitz, E.: Principles of mixed-initiative user interfaces. Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99. pp. 159-166. ACM Press, New York, New York, USA (1999).
4. Ojala, T., Kostakos, V., Kukka, H., Heikkinen, T., Linden, T., Jurmu, M., Hosio, S., Kruger, F., Zanni, D.: Multipurpose interactive public displays in the wild: Three years later. IEEE Computer. 45, 42-29 (2012).
5. Hosio, S., Jurmu, M., Kukka, H., Riekki, J., Ojala, T.: Supporting distributed private and public user interfaces in urban environments. Proc. HotMobile 2010, Annapolis, MD, USA, 25-30. pp. 25–30. ACM, New York, NY, USA (2010).
6. Davies, N., Friday, A., Newman, P., Rutlidge, S., Storz, O.: Using bluetooth device names to support interaction in smart environments. International conference on Mobile systems applications and services Mobisys 09. pp. 151-164. ACM, Kraków, Poland (2009).
7. Churchill, E., Girgensohn, A., Nelson, L., Lee, A.: Blending digital and physical spaces for ubiquitous community participation. Communications of the ACM. 47, 38 (2004).
8. Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M.: Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. Proceedings of the 16th annual ACM symposium on User interface software and technology - UIST '03. pp. 159-168. ACM Press, New York, New York, USA (2003).
9. Vogel, D., Balakrishnan, R.: Interactive public ambient Displays: Transitioning from Implicit to Explicit, Public to Personal, Interaction with Multiple Users. Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04. p. 137. ACM Press, New York, New York, USA (2004).
10. Morales-Aranda, A.H., Mayora-Ibarra, O.: A Context Sensitive Public Display for Adaptive Multi-User Information Visualization. Third International Conference on Autonomic and Autonomous Systems (ICAS'07). p. 63 (2007).
11. Peltonen, P., Kurvinen, E., Salovaara, A.: It's Mine, Don't Touch!: interactions at a large multi-touch display in a city centre. Proceedings of CHI2008. pp. 1285-1294 (2008).
12. Sacks, H., Schegloff, E.A., Jefferson, G.: A Simplest Systematics for the Organization of Turn-Taking for Conversation. Language. 50, 696-735 (1974).
13. Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A., Lopes, C.: Making sense of sensing systems. Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02. p. 415. ACM Press, New York, New York, USA (2002).