Hindawi Publishing Corporation
Advances in Human-Computer Interaction
Volume 2014, Article ID 371867, 17 pages
http://dx.doi.org/10.1155/2014/371867

Research Article

Interaction Tasks and Controls for Public Display Applications

Jorge C. S. Cardoso[1] and Rui José[2]

[1] *CITAR/School of Arts, Portuguese Catholic University, 4169-005 Porto, Portugal*
[2] *Algoritmi Centre, University of Minho, 4800-058 Guimarães, Portugal*

Correspondence should be addressed to Jorge C. S. Cardoso; jorgecardoso@ieee.org

Received 2 November 2013; Revised 18 February 2014; Accepted 18 March 2014; Published 10 April 2014

Academic Editor: Thomas Mandl

Public displays are becoming increasingly interactive and a broad range of interaction mechanisms can now be used to create multiple forms of interaction. However, the lack of interaction abstractions forces each developer to create specific approaches for dealing with interaction, preventing users from building consistent expectations on how to interact across different display systems. There is a clear analogy with the early days of the graphical user interface, when a similar problem was addressed with the emergence of high-level interaction abstractions that provided consistent interaction experiences to users and shielded developers from low-level details. This work takes a first step in that same direction by uncovering interaction abstractions that may lead to the emergence of interaction controls for applications in public displays. We identify a new set of interaction tasks focused on the specificities of public displays; we characterise interaction controls that may enable those interaction tasks to be integrated into applications; we create a mapping between the high-level abstractions provided by the interaction tasks and the concrete interaction mechanisms that can be implemented by those displays. Together, these contributions constitute a step towards the emergence of programming toolkits with widgets that developers could incorporate into their public display applications.

1. Introduction

Public digital displays are becoming increasingly ubiquitous artefacts in the technological landscape of urban spaces. Many of those displays are also becoming more interactive, enabling various forms of user engagement, such as playing games, submitting photos, or downloading content. In general, interaction is clearly recognized as a key feature for public displays in both the research literature and commercial systems, and a very broad range of interaction techniques have been proposed to create all sorts of interactive display systems. Still, even if creating a particular interactive solution is not in itself a major technical challenge, the approaches used are essentially ad hoc solutions that are specific to one particular system and interaction experience. The problem is that there are no abstractions for incorporating interactivity into public display applications that may help interaction support become a commodity in public displays. As summarized by Bellucci et al. [1], *"At present, there are no accepted standards, paradigms, or design principles for remote interaction with large, pervasive displays."*

The fundamental reason why this happens is because display systems are still based on proprietary technology and displays networks are operated as multiple isolated islands, each with its own concepts and technologies. We envision public displays to progressively move away from a world of closed display networks to scenarios in which large-scale networks of pervasive public displays and associated sensors are open to applications and content from many sources [2]. In these scenarios, displays would become a communication medium ready to be appropriated by users for their very diverse communication goals. Third-party application developers would be able to create interactive display applications that would run across the many and diverse displays of the network and interaction would necessarily become an integral part of the whole system.

However, the current lack of interaction abstractions represents a major obstacle to this vision and to the widespread adoption of interactive features in public displays, for both application developers and users. For developers, this means that they all have to develop their own approach for dealing with a particular interaction objective using a particular

interaction mechanism, leading to extra development effort outside of the core application functionality. In addition, each developer replicates this effort, potentially originating poor designs and wasted effort. For users, the lack of well-known interaction abstractions is also a problem, as they need to deal with inconsistent interaction models across different displays. Without familiar abstractions people are not able to use their previous experiences to develop expectations and practices regarding interaction with new public displays.

There is a clear analogy between these problems and the early days of the graphical user interface, when desktop computer programmers had to make a similar effort to support their interaction with users. The problem was addressed with the emergence of reusable high-level interaction abstractions that provided consistent interaction experiences to users and shielded application developers from low-level interaction details, as in the XToolkit [3]. Nowadays, when developing desktop applications, developers can focus on the interaction features of their applications and abstract away from low-level issues, such as receiving mouse pointer events, recognizing a click on a specific button or changing the visual state of a button that has just been clicked. These low-level input events are encapsulated by user interface widgets that provide developers with high-level interaction abstractions, thus facilitating the task of creating an application. From the usability perspective, widgets also enforce consistency of the interface, allowing users to learn to interpret their affordances in a way that enables them more easily to tackle new interfaces and programs by building on their previous experience.

This type of abstractions may now also provide an important inspiration for addressing the similar problem being faced by public displays, where the transition to a new era of generalized interaction support will also require a step up in the abstraction scale. As described by Mackinlay et al. [4] *"to achieve a systematic framework for input devices, toolkits need to be supported by technical abstractions about the nature of the task an input device is performing."* The proliferation of input devices and techniques for public displays reached a point at which it is both possible and fundamental to systematize the knowledge that may support the design of interaction toolkits for public display systems and ultimately enable interaction to become a common element of any display application in open display networks.

Our objective is to take a first step in that direction by uncovering interaction abstractions that may lead to the emergence of interaction controls for applications in public displays. Given the broad diversity of public display systems and interactions models, any solid contribution in this area needs to be anchored on a clear identification of the main assumptions being made about the nature of the displays and the interactions they aim to support. In our work, we assume an interaction context in which large shared displays are being used as the execution environment for multiple applications, each with potentially various concurrent users that interact with them through various interaction modalities, for example, Bluetooth, SMS, or visual codes. We also assume that interactions are based on a shared model of appropriation in which no single user can be expected to fully appropriate the main public display at any moment. This considerably reduces the applicability of our work to displays where individual appropriation is normally assumed, such as those based on touch or gesture-based interfaces.

To reach our goal, we have made an extensive review of 52 publications about interactive public display systems and coded the description of their interaction features. The codes generated from that process were then aggregated into major interaction tasks, a concept borrowed from Foley et al. [5], each with its own properties and possible values for those properties. We then matched these interaction tasks against the concrete interaction mechanisms identified in the literature, plotting the various implementations found on the literature in a spatial layout of a design space that extends previous work by Ballagas et al. [6]. Finally, we explored different combination of properties and values associated with the interaction tasks and outlined a set of concrete interaction controls that can provide a starting point for the development of interaction toolkits for interactive public display applications.

The novel contributions of this work are a new set of interaction tasks focused on the specificities of public display interaction, a characterization of interaction controls that may enable those interaction tasks to be integrated into applications for public displays and a mapping between the high-level abstractions provided by the interaction tasks that have been identified and the concrete interaction mechanisms that can be supported by public displays. Together these contributions constitute a step towards the emergence of programming toolkits with interaction controls that developers could incorporate into their public display applications.

2. Abstracting Interaction

As a first step in our research, we have made a more in depth analysis of the concept of interaction abstraction and particularly to what extent the approaches from the desktop domain could be applied to this new domain of public displays.

2.1. Revisiting Desktop Abstractions. In the early days of graphical user interfaces, application developers were facing a similar problem as the one currently posed by public displays, as there was not a consistent way to integrate interaction features—*actions interpreted in the context of the application's semantic domain, provided by the application to a user*—into the applications. This was addressed with the emergence of various conceptual frameworks for interaction, such as pointer based graphical interaction.

Mackinlay et al. [4] proposed a design space of input devices, using a human-machine communication approach. In their design space, they consider the human, the input device, and the application: the human action is mapped into parameters of an application via mappings inherent in the device. *"Simple input devices are described in terms of semantic mappings from the transducers of physical properties into the parameters of the applications."* A device is described as a six-tuple composed of a manipulation operator, input domain of possible values, a current state, an output domain, and additional device properties. This six-tuple can be represented

diagrammatically, and this graphical representation of the design space has been used extensively to characterize and compare different input devices.

Foley et al. [5], produced a taxonomy which organizes interaction techniques around the interaction tasks they are capable of performing. *The interaction tasks represent high-level abstractions that essentially define the kind of information that applications receive in result of a user performing the task*. They form the building blocks from which more complex interactions, and in turn complete interaction dialogues, can be assembled. They are user-oriented, in that they are the primitive action units performed by a user. Foley's tasks were based on the work by Deecker and Penny [7] which identified six common input information types for desktop graphical user interfaces: position, orient, select, path, quantify, and text entry. Foley also identified various interaction techniques that can be used for a given task and discussed the merit of each technique in relation to the interaction task. In this work, we use the concept of interaction task as defined by Foley to analyse interaction with public displays.

Myers [8] proposed *interactor* objects as a model for handling input from the mouse and keyboard. An interactor can be thought of as an intermediary abstraction between Foley's taxonomy and concrete graphical user interface (GUI) widgets. Interactors support the graphical subtasks, but abstract the concrete graphics system, hide the input handling details of the window manager, and provide multiple behaviours, such as different types of graphical feedback that can be attached to user interface objects. Myers defined six interactors: menu-interactor, move-grow-interactor, new-point-interactor, angle-interactor, text-interactor, and trace-interactor. The same interactor can be used to implement various concrete GUI widgets.

This type of research led to the now widely used concept of *user interface widget* (also known as "interaction objects," "controls," or simply "widget"): *an abstraction that hides the low-level details of the interaction with the operator, transforming the low-level events performed by the operator into higher level events*—Bass and Coutaz [9]. Widgets provide support for the three main stages of the human action cycle [10]: goal formation, execution, and evaluation. Their graphical representations and feedback support mainly the goal formation and evaluation stages. Widgets have a graphical representation that application developers use to compose the graphical user interface (GUI) of the application, supporting users in the goal formation stage by providing graphical representations to the interaction features of an application. Widgets also support the evaluation stage by providing immediate graphical feedback about their state. For example, a textbox widget echoes the typed characters to show what users have already written and shows a blinking text cursor to indicate that it can accept more input. The internal behaviour of a widget supports the execution stage and insulates applications from low-level input events transforming them into high-level interaction events. For example, an application that needs users to input a text string does not need to handle individual key presses; it can use a textbox widget that does this low-level handling and passes back to the application the complete text string. In widget toolkits, interaction events are usually defined as asynchronous function calls made by the interaction software system to the application. The kind of information carried by the interaction event defines what interaction task is being accomplished. From an informational perspective, multiple types of widgets could be used to accomplish a desired task. For example, to allow users to input a number, programmers often have at their disposal several types of data entry widgets—number type-in boxes, sliders, and spinners—that can restrict the type of accepted data and provide different interaction events (a type-in box usually triggers an event only after the number is entered, while a slider fires a sequence of events with intermediate values as the user drags the slider). Even though our work is strongly inspired by the widget metaphor, in this paper we use the more general term control to designate the same kind of abstraction. This is because widgets have a strong connotation with a particular graphics and interaction paradigm that may not be appropriate for public displays.

There are also model-based user interface development languages and tools that provide support and useful abstractions for various phases of the software development cycle. For example, the MARIA language [11] is a model-based user interface description language targeted at applications for ubiquitous environments. Interactive public display applications, however, are not yet mature enough for the emergence and use of model-based tools. By characterizing the interaction tasks and controls that are suitable for public display interaction, our work may help to consolidate the level of abstraction needed to successfully use model-based tools and languages.

2.2. Interaction in Public Displays. While it seems reasonable to apply successful lessons from the desktop world, there are significant differences that need to be accounted for when considering the adaptation of those principles to the specifics of the interaction environment around public displays. Using the concept of "ecosystem of displays" introduced by Terrenghi et al. [12], we could generally describe the public display environment 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"* [12]. Although there can be many kinds of social interaction in these spaces, we are focusing essentially on many-many interactions where there is not a single person or small group that "owns" the information of a display. Instead, the aim is to create a shared information space where everyone can have the same opportunities to interact and where the different displays offer different views to the information or different possibilities to interact with it. The different sized displays afford different types of interaction but they can function in an integrated way in the ecosystem, offering different synergies and opportunities [13]. For example, touch-sensitive surfaces in the tables of a bar or the personal devices of people in that same bar may all be used as privileged input devices to a public display system for sharing content on a larger vertical public display. Multiple users may generally share these public display systems at the

same time, even if in a noncoordinated way, interacting with the various features of the system, using different interaction mechanisms, both remotely (e.g., using a mobile device) and at close distance (e.g., touching the public display itself).

Unlike desktop systems, which usually rely on a very small set of input devices—most often just a keyboard and mouse—public display interaction can take advantage of very different interaction mechanisms. For example, Ballagas et al. [14] have proposed two mechanisms that make use of camera-phones to interact with public displays: the sweep technique, where the camera-phone is used as a mouse with the optical flow determining the amount and direction of movement from sequential images taken by the phone's camera and the point and shoot technique, where an overlay of visual codes on the public display is used to allow the phone to determine the absolute coordinates of the point the camera is pointing at. Bluetooth naming [15, 16] has also been used as an interaction mechanism by providing a simple command language that users can use in the names of their Bluetooth devices, which are continually scanned and evaluated by the display system. Bluetooth file exchanges between users' devices and the display system has also been explored, for example, by Cheverst et al. [17] in the Hermes Photo display system. Dearman and Truong [18] have proposed a DTMF (dual-tone multifrequency signalling) based solution for interacting with public displays where users can control applications by connecting their phone to the display system via Bluetooth and pressing keys on the mobile phone that are mapped to different actions on the application. Many other input mechanisms such as SMS/MMS [19], email and instant messaging [20], Twitter [19], RFID tags [21], gestures [22], and face detection [23], to name a few, have been explored for public display interaction. The UBI-Oulu infrastructure is a relevant example of a multiapplication network of public displays that offers a wide range of services via various interaction modalities [24], including a $57''$ capacitive touch screen, two overhead cameras, an NFC/RFID reader, and Bluetooth. A number of web-based interactive applications can be accessed through an application menu and used through a combination of the interaction modalities.

The breadth of mobile interaction mechanisms has already motivated research that tries to systematize the cumulative knowledge around mobile techniques for interaction. Building on Foley's interaction tasks, Ballagas et al. [6] developed a design space for comparing how different mobile device based input techniques could support a given interaction task. The input techniques were compared along various dimensions such as the number of physical dimensions (1d, 2d, 3d), the interaction style supported, the type of feedback provided, and whether the technique provides absolute or relative values. As stated by the authors, their design space is *"an important tool for helping designers [...] select the most appropriate input technique for their interaction scenarios."* The work by Ballagas et al. [6] provides a valuable design space for reasoning about the multiple types of interaction with public displays using mobile devices. We thus used this as a starting point for our own work and extended it in two ways: by considering not just the smart-phone, but also other interaction devices and by considering the existence of new interaction tasks, beyond the ones defined by Foley et al., which may give a broader and more specific view of the interaction space with public displays.

3. Interaction Tasks for Public Displays

To uncover interactive tasks for public displays, we made a comprehensive study of existing publications around the topic of interactive public displays. This approach aimed to go beyond specific interaction techniques and allow common interaction patterns to emerge from the assumptions and approaches applied across a broad range of interactive display systems. Our research followed an approach based on the grounded theory methodology [25], borrowing many of its phases: open, selective, and theoretical coding; memoing; and sorting.

We started with an initial set of 12 papers and did a first phase of open coding, in which we produced our first set of codes corresponding to specific attributes of the respective interactions. We then analysed these codes to aggregate some of them and remove others that were deemed not relevant from the interaction point of view. This much smaller set of relevant codes was used as the starting point in a second coding phase, where we coded 40 additional papers. These additional papers were selected from standard academic services (ACM Digital Library, IEEE Xplore Digital Library, Google Scholar) based on keyword searches for interactive public displays. We further refined the paper selection task to guarantee a balanced combination of various interaction mechanisms, various application domains, and various types of displays. This paper selection process was iterative and simultaneous to the coding procedure. Following upon Ground Theory principles, we continued to select new papers until the coding was saturated. Simultaneously, we started a third theoretical coding phase, identifying relationships between the existing codes and producing new codes to reflect these relationships. In this phase, we started organizing the existing codes into categories of interactions, along with their properties and concrete values associated with those properties. We adopted the definitions of categories and properties from Glaser and Strauss [25]: *"A category stands by itself as a conceptual element of a theory. A property, in turn, is a conceptual aspect or element of a category."*

To identify and distinguish categories, we analysed the interaction features that were being described, based on the underlying types of information that had to be exchanged between the user and the display system. These second and third phases were highly iterative and intermixed: we recoded previously coded papers more than once to make sure their coding was up-to-date with the latest categories and properties. For example, if we identified a new property while coding a paper, we would go back to previous publications and make sure we coded that property, in case we had missed it originally. The complete process originated a total of 87 codes that referenced 448 text segments in the 52 papers [26].

Memoing, that is, writing ideas associated with codes, was also an important part of the methodology and this went in

Table 1: Interaction tasks, properties, and values.

Task	Property	Values
Select	Type of selection	[Action, option]
Data entry	Bounds	[Unbounded, bounded]
Upload	Media type	[Text, image, video, audio, etc.]
	Media location type	[Personal device, public location]
Download	Media type	[Text, image, video, audio, etc.]
	Media location type	[Display system, public location]
	Target device	[Smartphone, email, USB stick, print]
	Target user	[Self, other]
Signal presence	Location disclosure	[Automatic, manual]
	Location verification	[Verified, unverified]
Dynamic manipulation	Type of manipulation	[Cursor, joystick, keyboard, skeleton/silhouette]

parallel with all the coding phases. We used memos to start relating our codes together and forming a structured view (of categories, properties, and values) of all the interaction tasks that were emerging. We also used memos to note possible missing properties and values that we needed to search in additional publications to make sure our categories were saturated. The memos associated with the categories became the first raw descriptions of our interaction tasks in the final description and analysis, after we sorted them to chain the ideas that emerged during the coding phases and turn them into a more logical narrative.

The categories that resulted from the coding process correspond to the interaction tasks that define the general information that the application needs to specify and the information that the application receives in the interaction events. The interaction tasks have properties that can take different concrete values and restrict the information or the behaviour associated with the task. These properties and values of the interaction tasks are mapped directly from the properties and values that resulted from the coding process. For example, the passage *"CoCollage users who are connected to the web site in the café may also send messages directly to CoCollage via a textbox near the upper right of any page*" is describing an interaction feature that allows users to send a text message to the display. In the third coding phase, this feature was coded with "data entry" (category), "bounds" (property), and "text" (value).

The result of this analysis is the list of 6 interaction tasks summarised in Table 1. As previously defined in Section 2.1, these interaction tasks should be seen as representing the main types of information exchange that may occur between the system and a user as part of an interactive event. They are essentially low-level tasks that focus on interaction itself and are not meant to represent high-level user goals, as is normally the case in the context of task analysis and modelling.

We will now describe in more detail each of those interaction tasks, characterizing them in terms of the respective information exchange, the associated properties, and the possible values for those properties. Whenever appropriate, we will illustrate these properties with specific examples from the surveyed display systems.

3.1. Select. The select task is equivalent to the select task of Foley et al. [5], allowing users to trigger actions or select options in an application. It requires applications to specify the complete set of options or actions they wish to provide to users. The interaction event triggered by the display system will include the action or option identification, so that the application can determine which one was selected.

3.1.1. Type of Selection. The *type of selection* property refers to what users are selecting: an action to be triggered immediately by the application or an object from a list of possible objects. Using the terminology of Cooper et al. [27], in action selection users input a verb (what action the application should perform), and the noun (the object on which to act) is usually implicit. In object selection, users input a noun and later a verb (or the verb is implicit). These two types of selection are traditionally represented graphically in very different forms; for example, on desktop systems programmers usually have at their disposal different sets of widgets for triggering actions (menus, toolbars, buttons) and for selecting objects (list boxes, dropdowns).

In regard to triggering actions, Vogel and Balakrishnan [22] in the Interactive Public Ambient Displays system provide an example using hand gestures: *"Two complimentary hand postures are used to hide and show the display of a user's own proxy bar. The hide action is performed with a palm away posture consisting of an open hand pointing up with palm facing the display [...], analogous to the commonly seen* "stop" *gesture used for traffic signalling in real life."* QR codes are also a common alternative to provide users with a visual representation for an action, whether in a live public display or printed on paper. In the Mobile Service Toolkit/Mobile Service Explorer (MST/MSE) [28], for example, users could scan a visual code to have access to various actions: *"Sally can click on the tag using her MSE-enabled phone to establish a Bluetooth connection with the service. As soon as the phone connects with the service, her phone displays a message containing the current queuing time and asks whether she'd like to join the queue."* Another example is the Bluetone system [18], where users can use their phone's keypad to issue commands: *"[...] a user is able to watch a particular YouTube video, but also has the added ability of controlling audio/video*

playback. The user presses '5' on their mobile phone to pause the video [. . .]."

Selecting an object or item from a set of related items is also a frequently used feature, as the following examples show. The e-Campus system [16] provided a Bluetooth naming based interaction mechanism for selecting a song to play: "*By subsequently changing their device name to '\ec juke <song id>' the selected music track will be added to the queue of songs to be played.*" In this case users explicitly enter the action to be performed (i.e., "juke") and the item on which the action should take place (i.e., the song id). More often, the action is implicit and users just need to select the item to be acted on from a list presented by the public display, as in this Plasma Poster [29] example, which used a touch-screen interface: "*. . . this was the last item posted to the Plasma Poster Network, and the display cycle is about to begin again. Readers can select any thumbnail to be displayed by pressing it.*" SMS is also frequently used for this purpose, as in LocaModa's Polls [19] application were users would vote by selecting a choice from a list presented by the display.

3.2. Data Entry. The data entry task allows users to input simple data (text or numeric data) into a public display. Applications need to specify which type of data they wish to receive (text, numeric, dates, etc.) and possible bounds or patterns on the values they can accept. The interaction event that the application receives carries the user-submitted data. The data entry task is equivalent to the combination of the "quantify" and "text entry" tasks defined by Foley et al. [5]. In their classification of desktop applications' controls, Cooper et al. [27] combine quantify and text entry controls into a single category of "data entry." In our analysis, we also chose to combine them because if we abstract the interaction paradigm and consider the information exchange between user and application, quantifying and entering text are essentially the same: users input values into the application.

3.2.1. Bounds. The *bounds* property of the data entry task refers to whether the application accepts free text from the user or whether it imposes some predefined format to the data. For example, integer number within a limit or text that corresponds to a valid email address. This is an important property to consider because it imposes restrictions on the possible interaction mechanisms that can be used. For example, there is no easy way to guarantee that a value entered via an SMS message, Bluetooth naming, email, or other text-based interaction mechanisms conforms to the format accepted by the display system, before the user sends it. Special care should be taken if these mechanisms are used for bounded data entry as they can result in user frustration if the display system rejects the input.

Unbounded text entry corresponds to Foley's text entry task, in which users are allowed to submit a string of text that does not need to conform to any specific rule. Text entry can be used to send messages, comments, and keywords, to the public display. In CoCollage [30], for example, users could send messages to the display by entering text in the display systems' web page. Entering search terms is also a common use of text entry feature in public displays. Davies et al. [16], for example, provided a Flickr search application in their e-Campus system: "*Users can access photos on Flickr by changing their [Bluetooth] device name to 'ec flickr <search term>'. For example 'ec flickr oranges' would cause photos retrieved using the search term 'oranges' to be displayed.*" SMS is also frequently used to allow user input. LocaModa's Jumbli application [19], for example, allows users to play a word game by texting their words. Touch interfaces can also be used for these interactions, supporting the traditional desktop entry controls such as sliders and dials but also text-entry via onscreen keyboards as in the Digifieds [31] system.

Bounded data entry restricts the type, pattern, and range of the values that are entered. For example, in Visual Code Widgets, Rohs [32] described how visual widgets could be used with a camera phone: "*Unlike free-form input widgets, which provide 'unbounded' input, sliders are 'bounded' data entry widgets. The slider can be moved across a certain range, the selected value being proportional to the current slider position. [. . .] there are horizontal and vertical sliders. Input can either be continuous or discrete.*" Rating is another example of a bounded entry control, which usually allows users to enter a 1–5 value for an item. In CWall [33] users could rate the content items presented by the public display by touching an icon near the item. Bluetone [18] also allowed users to input bounded numeric values, in this case, using the mobile phone's keyboard.

3.3. Upload Media. The upload task allows applications to receive media files sent by users. Applications should be able to specify the type of media they are interested in, but other parameters such as the maximum file size or maximum media duration (for video and audio) could also be of interest. The interaction event received just needs to specify the URL of the uploaded file.

3.3.1. Media Type. The *media type* property of the upload task indicates the type of media file being uploaded: image, video, audio, html, and many other types of office documents. In JoeBlogg [34], for example, images were used to create an artistic composition on the public display. In other cases, images were used as free-hand comments to existing content, creating a discussion thread, as in the Digital Graffiti project [35]. Audio and video are also often used media types. In the Dynamo system [36] for example, students could upload a variety of media files into the surface, including video and music files: "*During the two-week deployment, the use of Dynamo varied considerably: students displayed and exchanged photos, video and music, which they had created themselves or brought in from home [. . .].*"

3.3.2. Media Location Type. The *media location type* property of the upload task refers to the original location of the media. In many cases, the public display system accepts content that is stored in a personal device such as a mobile phone or even a USB pen drive. In these cases content is sent directly to the public display by attaching the pen drive or by transferring the file via Bluetooth OBEX or via a

custom mobile application. In the Hermes Photo Display [17], for example, users could transfer photos from their mobile phones to the display using OBEX: *"This version of the Hermes Photo Display also enables a user to [. . .] use her mobile phone's built-in 'picture' application in order to send a picture to the photo display over Bluetooth [. . .]."* In JoeBlogg, users would send personal pictures stored in their mobile phones via MMS. In Dynamo, users would simply attach their USB pen drives to the display to copy the media files into a shared space. Email has also been explored in the Plasma Posters [29] display system: *"Posted content can be images and movies (sent in email as attachments), formatted text and URLs."*

In other cases, however, users do not actually have a copy of the content in a personal device, but know the respective address. In these cases, the display receives a reference to the content, instead of the content itself. WebWall [37], for example, accepted URLs of media files to play in the public display: *"[. . .] there are other service classes that are better defined first over the Web-client: Video and picture galleries (service class Gallery) can be used to display multimedia content by composing URLs of the media to display [. . .]".*

3.4. Download Media. The download task allows users to receive a content item from the display and store it in a personal device or account for later viewing or reference. The interaction event received by the application can simply be an acknowledgement that the file was, or is about to be, downloaded.

3.4.1. Media Type. The *media type* property is analogous to the media type property of the upload task.

Just as in upload task, various media types may be provided by a display system and made available for users to download. The Hermes Photo Display [17], for example, allows users to *"[. . .] use the interface on the Photo Display to select a picture and then receive this picture onto her phone via Bluetooth."* Videos are also a common media type that users may want to download. In ContentCascade [38], for example: *"The display is playing trailers of upcoming movies. Bob sees the Shrek movie and decides 'I like that!' and wants to download the movie clip. He pulls out his Bluetooth enabled cell phone [. . .]."*

3.4.2. Media Location Type. The *media location type* property is analogous to its counterpart in the upload task: content to be downloaded can either be already publicly available and the display system just provides the address on the web or it can be content stored internally at the display system that is transferred to the user. For example, in ContentCascade users could also receive URLs in their mobile device, instead of the video itself. In Hermes Photo Display, however, the photos were stored internally in the display system and downloading involved establishing a Bluetooth connection between the display and user's mobile device to transfer the photo.

3.4.3. Target Device. The *target device* property refers to where the downloaded content is transferred as a result of the interactions. Downloaded media can be received in a variety of destination devices or personal accounts, using various communication protocols. Content can be downloaded to a personal mobile device, for example, using SMS as in LocaModa's Community Board application [19]. OBEX is another protocol that can be used for receiving media files as in the Hermes Photo Display. There are also examples of display systems that use custom mobile applications and communication protocols for receiving files on the mobile device. Touch and interact [39], for example, consists of a public display and a mobile application in which *"[. . .] the user interacts with a picture board by touching the picture with the phone and in response, the picture moves from the dynamic display to the phone."* Users can also receive files in a USB pen drive, as in the Dynamo [36] system or download to their mobile device by scanning a QR code as in Digifieds. Mobile devices, however, are not the only possibility for receiving media files. A popular approach is to allow users to receive the content in their email. In the Digital Graffiti project [35] for Plasma Posters, for example: *"Later, Jane is passing by the Plasma Poster and sees all the annotations that have been posted over her original content. She is amused to discover her post has caused so much response and debate and forwards the recommended URL to her home email so she can read it later."* Finally, a less common but also possible solution for specific media types it to allow users to print the content. Also in the Plasma Posters [29] project, users could print a displayed item directly from the public display.

3.4.4. Target Person. The *target person* property refers to whether the content is transferred to the interacting user or to another person. Often, users want to download content for themselves in order to get an offline copy of the content or as a reference to view later. However, there are also cases where a user wants to download a content item and forward it to someone else. In this case, users are effectively sharing content from the display. Plasma Posters, for example, allows content to be forwarded to others: *"Items can be forwarded to others, or to oneself for reading later at a personal computer."* The same could be done with Digifieds, which allowed users to send content via email: "*Digifieds can also be taken away [. . .] by sending them to an email address, [. . .].*"

3.5. Signal Presence. The signal presence task allows the application to be notified about events regarding the presence of users in the vicinity. Although all interactions with a display system can be used to determine the presence of users (if a button was pressed in a touch screen, it means that there was someone there), in this section, we are considering only those interactions specifically designed for determining the presence of users.

3.5.1. Location Disclosure. The *location disclosure* property refers to whether the user manually sets his presence, or whether the presence is sensed automatically by the display system. The manual form corresponds to a check-in interaction where users decide when they would like to announce their presence to the public display.

Check-in can be accomplished through a number of different ways, for example, using hardware that reads a

personal identity card or a personal mobile device. Magnetic card readers, RFID readers, or even Bluetooth detection can be used to accomplish this type of check-in. Russell and Gossweiler [40], for example, used personal cards that users could swipe on a card reader in the BlueBoard display to access their personal data (in this case, the feature worked as a login because it allowed access to personal information, but it could also be used for check-in): *"The net effect is that a user can 'log in' by simply swiping their badge at the display, getting rapid access to their content."* Check-in can also be accomplished solely through software, for example, through a mobile application or web page. CoCollage [30] provides a check-in feature to its users through a button on a web page. The automatic sensing of users around the display can itself be subdivided into three forms according to the level of information sensed: presence detection, characterization, and identification. Presence detection corresponds to an on/off detection where the display either detects someone (but not who or how many) in its vicinity or detects no one. This can be used to trigger a change in the display's mode from an ambient mode to a more interactive mode, as in the Aware Community Portals [23]: *"[...] a weather map triggered by the user walking by versus a news article shown when the user lingers to browse."* Presence characterization corresponds to a more rich detection, where the display is able to sense more information about the people in the vicinity, such as how many, their position, where they are gazing, and the estimated age. CWall [33] used computer vision techniques to infer if people were standing in front of the display and looking at it. In presence identification, the display is able to identify users and, possibly associating personal information. This can be used to provide personalized content on a public display as in the Proactive Displays [21]: *"When attendees are near a proactive display, content from their profiles can be shown."*

3.5.2. Location Verification. The *location verification* property indicates whether the system can verify that the user is really where he says he is. In the automatic presence sensing, the system can have stronger guarantees that users or at least their devices are in the vicinity of the display. Sensors are assumed to be located near the display, and they usually have a limited detection range. The same happens in the manual presence sensing that makes use of personal cards or other physical items that are detected by a card reader or other sensor near the display system. Even if the check-in is accomplished via software, the user's location can still be verified. CoCollage, for example, uses the local Wi-Fi network to verify the user's location: *"The presence of users is established via an explicit 'check-in' through the use of [...] a web page that is enabled only when the user's computer is connected to the wireless Internet router in the café."*

In many cases, however, the user's location is not verified by the system. Most location based social networks such as Foursquare, Google Latitude, and Facebook Places provide mobile applications that allow to check-in in any place, without any system verification about the real location of the user. This is something that is normally accepted by people as part of the semantics of presence through these check-in procedures. Some public display systems take advantage of these existing location based networks. LocaModa's Check-in application [19], for example, *"leverages widely adopted location based applications such as Facebook Places and Foursquare to display relevant venue Check-In activity on venue digital displays."* The Instant Places [41] display network provides its own mobile client with similar check-in semantics: *"Explicit session activation can be accomplished through a check-in mechanism available in our instant place mobile app."*

3.6. Dynamic Manipulation. The dynamic manipulation task corresponds to continuous interactions where users manipulate graphical objects in the application's interface. Dynamic manipulation represents tasks in which it is fundamental to provide a direct-manipulation style, particularly, "rapid, incremental, reversible operations whose impact on the object of interest is immediately visible" as described by Shneiderman [42]. In this task, the application receives a continuous, timely, flow of information, which it can then map to various graphical objects.

Although the dynamic manipulation task requires a direct manipulation interaction style, not all interactions in a direct manipulation style represent dynamic manipulation tasks. For example, the activation of a button, even if using some cursor-like interaction is still a selection task, as the application would only be interested in receiving an action selection event.

3.6.1. Type of Manipulation. The *type of manipulation* property refers to the type of action performed by the user and the information received by the application. We defined four values for this property: cursor, joystick, keyboard, and skeleton/silhouette input. Although their names may suggest physical devices, these types of input may be generated by highly diverse mechanisms (e.g., joystick input can be generated by a physical joystick, but also by specially arranged keyboard keys and even by a virtual multitouch joystick).

Cursor events carry information about the position and velocity of multiple cursors on a 2D or 3D environment and can be used for mouse, multitouch, or even 3D interactions. For example, Dynamo [36] allows users to "carve" rectangular regions on the display to appropriate them for individual use. This is done by simply "drawing" a rectangle using the mouse: *"Carves can be created by a mouse drag gesture to create privately owned areas in which only the user and their chosen members can interact."* In CityWall [43], users used multitouch gestures to move, scale, and rotate photos.

Joystick events carry information about the angle and state of joystick/gamepad buttons. In Point and Shoot [14] users could use their camera phone as a mouse or joystick and select rotate and move jigsaw puzzle pieces: *"The phone display is used to aim at a puzzle piece on a large display. [...] Pressing the joystick indicates selection and a visual code grid flashes on the large display to compute the target coordinates [...]."* In the Vodafone Cube [44], users could dial a phone number and control various games, including a car racing game, using the phone's keyboard as a joystick.

Table 2: Mapping between interaction tasks and touch-screen based interaction mechanisms.

Interaction mechanism	Interaction task					
	Select	Entry	Upload	Download	Presence	Dyn. Manip.
Touch screen	DDC: Plasma Posters; Hermes Photo Display; OutCast; Blueboard; Jukola; AgentSalon; Digifieds; Spalendar; iSchool; FizzyVis; Semipublic displays; CWall; UBI-hotspot; Vista	DCC: Plasma Posters; Digifieds; Spalendar; CWall		DDC: Plasma Posters; iSchool; CWall	DDC: UBI-hotspot	DCC: Plasma Posters; Blueboard; FizzyVis; CityWall; Semipublic displays
Touch screen + Bluetooth OBEX			IDC: Hermes Photo Display	IDC: Hermes Photo Display		
Touch screen + mobile application			IDC: Digifieds	IDC: Digifieds		
Touch screen + printer				DDC: Plasma Posters		

Keyboard events carry information about a succession of key presses in a physical or emulated keyboard. In MST/MSE [28], for example, the mobile client supported keyboard input: *"transmits all keypress events from the phone's keypad back to the MST server in real time."* Remote Commander [45] is another example where keyboard input was important: *"This allows [...] the PalmPilot [...] input to emulate the PC's keyboard input. The important point is that this works with all existing PC applications, without requiring any modifications to the applications themselves."* It should be noted that keyboard events do not necessarily mean that the application is interested in receiving text data (a keyboard could be used to play music, e.g.).

Skeleton/silhouette events carry information about the position of the user's body joints and/or about the user's silhouette. This type of input has recently gained wide exposure due to the Kinect depth camera controller, but it can also be accomplished with other sensor technologies such as body suits, stereo cameras, or motion capture systems. This kind of input has been mostly explored in artistic interactive projects, but it has also been applied successfully in public display systems. Müller et al. [46] in project Looking Glass, used a Kinect to extract user's silhouettes and provide a gaming experience in a public display of a shop window, by allowing users to wave their arms to push balls on the display.

4. A Design Space of Interaction Controls for Public Displays

Based on the interaction tasks described in the previous section, it is possible to frame a new design space for interaction with public displays around those tasks. In this section, we analyse how the interaction tasks could be mapped to interaction mechanisms and what interaction controls can be derived from them.

4.1. Mapping between Interaction Tasks and Mechanisms. The first step in our analysis is to explore the relationship between interaction mechanisms and the set of interaction tasks. This mapping provides a comprehensive view of how different mechanisms can be used to support given interaction tasks and also of how the various interactions tasks are represented in the various concrete system implementation from the research literature.

To facilitate the mapping, we created a spatial layout that shows how the different interaction tasks can be implemented with various interaction mechanisms. This mapping is inspired by the spatial layout from Ballagas et al. [6], but we omitted the attributes dimensionality and relative versus absolute, which were not relevant for our analysis, and we added a new interaction distance attribute. The resulting layout, depicted in Tables 2, 3, 4, and 5, represents how the interactive displays from the literature are distributed between the interaction tasks and the mechanisms that support those tasks. We have plotted each interaction mechanism that appeared in the surveyed interaction public display systems.

The reference to each interactive display system is complemented with a classification of the interaction along three secondary dimensions: interaction style, feedback, and interaction distance. The interaction style can be direct or indirect: *"in direct interactions, the input actions are physically coupled with the user-perceivable entity being manipulated, appearing as if there was no mediation, translation, or adaptation between input and output. In indirect interactions, user activity and feedback occur in disjoint spaces (e.g., using a mouse to control an on-screen cursor)"*—Ballagas et al. [6]. Feedback can be continuous or discrete: *"continuous interactions describe a closed-loop feedback, where the user continuously gets informed of the interaction progress as the subtask is being performed. Discrete interactions describe an open-loop feedback, where the user is only informed of the interaction progress after the subtask is complete"*—Ballagas et al. [6]. For the purpose of this analysis, we are only considering shared feedback shown on the public display itself and not the individual feedback that may be generated on the mobile

Table 3: Mapping between interaction tasks and interaction mechanisms based on personal mobile devices.

Interaction mechanism	Interaction task					
	Select	Entry	Upload	Download	Presence	Dyn. Manip.
Bluetooth detection					IDR: BluScreen	
Bluetooth naming	IDR: e-Campus	IDR: e-Campus; Instant Places; Bluemusic			IDR: Instant Places	
Custom mobile personal device	IDR: Pendle ICR: VisionWand	ICR: VisionWand			IDR: Pendle; AgentSalon	ICR: VisionWand
DTMF						ICR: Vodafone Cube
DTMF + Bluetooth mobile phone	IDR: Bluetone	ICR: Bluetone				ICR: Bluetone
MMS			IDR: JoeBlogg			
Mobile application	DDR: Jukola DCR: C-Blink IDR: Mobilenin	IDR: Digital Graffiti; Hello.Wall; CWall; Mobile Service Toolkit	DDR: C-Blink IDR: Digital Graffiti; CWall	DDR: C-Blink IDR: Hello.Wall; Mobile Service Toolkit	IDR: Hello.Wall; Mobile Service Toolkit	ICR: Digital Graffiti; Remote Commander; Mobile Service Toolkit
Mobile application + Bluetooth mobile phone			IDR: Publix	IDR: Content-Cascade; Publix	IDR: Publix	ICR: Publix
Mobile application + camera phone	ICR: Sweep					DCR: Sweep ICR: Jeon et al.
Mobile application + camera phone + visual codes	DDR: Point & Shoot IDR: Visual code widgets; Mobile Service Toolkit	IDR: Visual code widgets		DDR: Digifieds		ICR: Jeon et al.
Mobile application + NFC phone + NFC display	DDC: Touch & Interact; Hello.Wall DDR: Hello.Wall		DDC: Touch and Interact	DDC: Touch and Interact	IDC: Touch and Interact	DCC: Touch and Interact
Personal id card					DDC: Blueboard; UBI-hotspot IDR: CoCollage; Proactive displays; GroupCast	
SMS	IDR: LocaModa	IDR: WebWall; LocaModa	IDR: LocaModa			

device, for example, which may be considerably more flexible. In the interaction distance we distinguish between close-up and remote interaction. Close-up interaction requires users to touch the display with their body (often fingers and hands) or with a hand-held device, whereas in remote interaction users can interact at a distance. This dimension has implications on the physical placement of the public display (close-up interaction require displays that are at arms reach) or on which interaction mechanisms are suitable for an already deployed public display. Each entry in the table is labelled with an ordered set of letters corresponding to the possible values for the three dimensions: Direct/Indirect, Continuous/Discrete, and Close-up/Remote.

We now use Tables 2 through 5 to analyze how four common categories of interaction mechanisms—touch-screen based public displays, interaction via mobile devices, device-free interaction, and desktop-like interaction—can be used to support the various interaction tasks, using concrete examples from the design space.

4.1.1. Interaction Based on Touch-Screens. Touch-screens can be used without the need for any other device so they are a good solution for walk-up-and-use, close-up interaction displays, provided that they can be placed in a location that allows users to directly touch it. Touch-screens can be used

Table 4: Mapping between interaction tasks and device-free interaction mechanisms.

Interaction mechanism	Interaction task					
	Select	Entry	Upload	Download	Presence	Dyn. Manip.
Camera	IDR: Aware Community Portals	ICR: Beye and Meier			IDR: Aware Community Portals; ReflectiveSigns; CWall; CWall; UBI-hotspot; SmartKiosk	
Camera (Kinect)	DCR: MAID				IDR: Code Space	DCR: Looking Glass; Code Space; MAID
Camera (MoCap system)	IDR: Interactive Public Ambient Displays; Spalendar	ICR: Spalendar; Spalendar			IDR: Interactive Public Ambient Displays; Spalendar	DCR: Spalendar ICR: Interactive Public Ambient Displays; Spalendar
Electromagnetic sensor (gestures)	ICR: Gesture Frame					ICR: Gesture Frame
Sound (finger click)	IDR: Gesture Frame					

Table 5: Mapping between interaction tasks and desktop-like mechanisms.

Interaction mechanism	Interaction task					
	Select	Entry	Upload	Download	Presence	Dyn. Manip.
Desktop application	ICR: Notification Collage	ICR: Notification Collage		ICR: Notification Collage		ICR: Notification Collage
Email	IDR: WebGlance	IDR: Locamoda; WebGlance	IDR: Plasma Posters; CWall	IDR: Digifieds		
Instant messaging	IDR: WebGlance	IDR: WebGlance				
Mouse and keyboard	ICR: Dynamo; Opinionizer	ICR: Dynamo; Opinionizer				ICR: Dynamo
Mouse and keyboard + USB stick			ICR: Dynamo	ICR: Dynamo		
Web application	ICR: CoCollage	IDR: CoCollage	IDR: CoCollage; WebWall; Digifieds	IDR: Digifieds	IDR: CoCollage	

to support most of the interaction tasks for public displays. Select, entry, and dynamic manipulation tasks are obviously well supported. Download media can be accomplished in a limited way by forwarding the content to a personal email address entered using a virtual keyboard or by selecting a username from a list in case the display system has registered users. Signalling presence can be supported in a manual way as in the Ubi-hotspot [47] system where users can touch the display to trigger a transition from a passive mode to an interactive mode. None of the public display systems we surveyed used a touch-screen (without any other device) for uploading media, although one could conceive that it could be used for uploading by entering the public address of a file using a virtual keyboard. However, touch-screens in conjunction with other devices can provide richer interactive experiences and better support for the full range of interaction tasks. The download and upload tasks in particular can take advantage of personal mobile devices for an easier transfer of media files by using an approach similar to the one used by the Hermes Photo Display with Bluetooth OBEX transfers or the Digifieds approach with visual and textual codes. Signalling presence can also be made more flexible by incorporating personal card readers into the display as in the BlueBoard or Ubi-hotspot display systems.

4.1.2. Interaction Based on Personal Mobile Devices. Remote interaction can be accomplished through many interaction mechanisms. A popular approach is to provide a custom mobile application (usually for smartphones) for interacting with the display. Some mobile applications require specific mobile hardware to function properly, such as having a camera, Bluetooth, infrared, and NFC; other mobile applications require the display to be able to generate visual codes. Most of these mobile applications provide an indirect interaction style with the public display where the user's focus is on the mobile device interface. Some, however, turn the mobile device into a tracked object as in C-Blink and Point and Shoot or into a viewport into the public display interface, as in the Visual Code Widgets, which provides a direct interaction style but

also requires users to stand closer to the display and hold the device in front of it. These solutions cover the complete set of interaction tasks for public displays, allowing users to have a rich interaction experience with a public display, remotely.

Another frequent alternative is to use the standard processing and communication features of mobile devices, without the need to install additional applications. Bluetooth detection, Bluetooth naming, SMS, Multimedia Message Service (MMS), and Dual-Tone Multifrequency (DTMF) have been used to support different interactive features. Although these interaction mechanisms do not support all the interaction tasks, they may still be a viable solution for specific interactions. Bluetooth has the advantage of being widely supported by mobile devices and cost-free for the user. Bluetooth detection, that is, scanning the area near the public display for Bluetooth enabled devices and reading their ids, can be used to estimate the number of people that are present and to determine which devices have been near the display and when, as in the BluScreen system. SMS and Bluetooth naming, that is, interpreting the Bluetooth name of the device as commands to the display system, can be used for selection and data entry, even if in a simple way, as in the e-Campus, and Instant Places systems. MMS can be used to upload or download pictures and other media files. The downside of both SMS and MMS is that it requires users or display system to incur in costs (which can be considerable for MMS) when sending the messages. Finally, DTMF can be used to support selection and data entry tasks as in the Bluetone system and dynamic manipulation as in the Vodafone Cube. DTMF also has costs for users, unless it is done over Bluetooth as in Bluetone.

4.1.3. Device-Free Interaction. Device-free interaction with public displays can be accomplished with cameras (standard web cameras or depth sensing cameras such as the Kinect) and computer vision techniques. Device-free interaction has the advantage of providing a walk-up-and-use interaction and not requiring users to directly touch the display, allowing it to be positioned in a way that allows multiple users to see and interact with it simultaneously. With devices such as the Kinect, it can be a viable solution in scenarios such as shop windows where it can also be used to detect and attract passers-by. Selection, data entry, presence, and dynamic manipulation tasks can be accomplished with these interaction mechanisms. Although device-free interaction by itself does not support download and upload tasks, it is possible to use additional devices for this purpose as in Bragdon et al. [48].

4.1.4. Desktop-Like Interaction. It is also possible to support all the interaction tasks through desktop-like interfaces. One possibility is to provide a custom native or web application that enables users to interact with the public display. All the interaction tasks can easily be supported in this manner. For example, Notification Collage, CoCollage, and Digifieds provide applications that mediate the interaction with the public display itself. It is also possible to provide a desktop-like interaction where the public display application itself behaves in a similar manner to a desktop application as in the Dynamo display where users simply pick up a mouse and keyboard to interact with the display. As in the case of mobile devices, it is also possible to use standard desktop applications such as email or instant messaging to interact with a public display system as in Plasma Posters, CWall, WebGlance, and other systems. Although it is not possible to support all the interaction tasks (e.g., dynamic manipulation is not possible with email or instant messaging), it can still be a plausible solution in some cases, as it leverages on existing applications thus obviating the need to install additional software.

4.2. Interaction Controls. Interaction controls provide the next element that is needed to enable applications to benefit from the interaction tasks that we have identified. The high level of abstraction that is associated with the interaction tasks needs to be instantiated into specific controls that can be integrated into applications to support interaction. A control can still maintain independence from the concrete interaction mechanism, but it refines the specific information being exchanged, defines additional optional and mandatory parameters, and can manage input in a specific way before triggering the interaction event. Just as we have several types of data entry controls for desktop applications, public display applications also need different controls for the same interaction task. These controls will form the main components that applications will use to provide their interaction features.

As part of our analysis of the interaction tasks, we sought to identify a representative set of controls that could illustrate how the various tasks could be instantiated. To define the set of controls we have considered the need to include all the interaction tasks, the key variations within each task and also what seemed to be the most common forms of interaction in the research literature, as illustrated in Tables 2 through 5. Still, this is not meant to be an exhaustive listing (As Foley et al. [5, page 20] put it *"their number is limited only by one's imagination"*), but it provides a good overview and comparison of the possibilities for implementing the various tasks for public display interaction and it should provide a relevant starting point for designing interaction systems for public display applications. The relevance of these specific controls will ultimately depend on their real world usage, which may lead to the emergence of totally new controls, changes to existing ones, and the disappearance of others.

In this description, we focus on the interaction events and information processing associated with the controls. We leave out the graphical representation and feedback aspects usually associated with widgets in desktop systems, as these would be very dependent on the specific implementation of the interaction system. Table 6 provides a list of possible controls for the various tasks.

Together, the mapping between the interaction mechanisms and interaction tasks and the characterization of the controls that support those tasks forms a design space for interaction abstractions for public displays that can be used in several ways. A designer of an interaction toolkit for public display applications can use the design space to understand the kinds of high-level controls that the toolkit

Table 6: List of possible controls for supporting the various interaction tasks.

Task	Control	Description
Select	Action	A generic action control, which causes the application to execute an action; similar to a desktop button. Triggers an event that identifies the action.
	Option list	A generic list control, which presents several options and allows users to select one (or more). Triggers an event with the selected option when a user makes a selection.
	Vote	Time based action control with a list of alternatives that waits for interactions during a predefined period of time. Triggers an event with the most voted alternative, when the time expires.
Data entry	Unbounded text	Allows users to input any string of text. Triggers an event with the input string.
	Bounded text	Supports various text patterns (such as email addresses, phone numbers, and dates). Triggers events with input string that conform to the specified pattern.
	Numeric entry	Generic numeric entry control allows users to input numbers, possibly with lower and upper limits, integer, or floating point. Triggers event with the input number.
	Rate	Allows users to rate content. May support various formats such as different scales, discrete/continuous rating scale. Triggers event with the input rating.
Upload media	Generic upload	An upload control that accepts any media file, possibly with a parameter to limit the total file size. Triggers an event with the location of the uploaded file.
	Video upload	Accepts only video files. Allows applications to specify the maximum duration of the video and supported video formats. Automatically converts between unsupported video formats to supported ones, for example, or simply does not allow unsupported formats.
	Image upload	Accepts only images. Allows applications to specify the maximum/minimum image size and supported image formats. Automatically converts images that do not conform to the specified size and format restrictions.
	Audio upload	Accepts audio files. Allows applications to specify the supported formats and maximum audio duration.
Download media	Download	Allows application to specify the media type and location of a content item that users can download. Triggers an event that identifies the downloaded file.
	Share	Allows users to share a content item with other people. Triggers an event that identifies the shared file.
Signal presence	Check-in	Allows users to explicitly signal their presence near a display. Optionally, the interaction event can carry the location verification status, allowing the display system to give more weight to check-ins with verified locations, for example.
	Presence	Signals the presence of users obtained implicitly from sensors. The information carried on the interaction event may vary, depending on the concrete types of sensors available, but we can generally categorize it according to the levels of information that are sensed: presence detection, characterization, and identification.
Dynamic manipulation	Cursor	Allows users to dynamically interact via (multiple) cursor positions. The interaction event is a continuous flow of cursor positions.
	Joystick	Provides joystick information (direction, gamepad button states), for gaming purposes. The interaction event is a continuous flow of direction and button states.
	Keyboard	Provides keystroke events. The interaction event is a continuous flow of key presses.
	Skeleton	Provides positioning of body joints and/or user's silhouette information (full or partial body parts). The interaction event is a continuous flow of skeleton/silhouette data.

should provide to application developers and which interaction mechanisms can support those controls. For someone deploying a public display system, the design space can be used to help make informed choices regarding the interaction mechanisms that should be deployed in order to support a specific set of interaction tasks. It can also be used to determine the interaction characteristics those interaction mechanisms impose. Application developers can use the design space to understand which controls can be supported by public display systems and decide how the interaction features of their applications can be implemented using those controls. Additionally, the various concrete examples of display systems listed in the design space can also be used as reference or design patterns, for the implementation of the various controls in the interaction toolkit.

4.3. PuReWidgets Toolkit. As part of our work on interaction abstractions for public displays, we have also created the PuReWidgets (Public, Remote Widgets) programming toolkit [49] for web-based interactive public display applications. This toolkit instantiates most of the interaction controls

that constitute our design space, enabling us to demonstrate the overall applicability of the design space in the context of the specific needs of multiple interactive applications for public displays.

We targeted web-based public display applications and created a programming library that developers of public display applications can incorporate into their applications to take advantage of widget-based interaction abstractions. In our current implementation, the programming library is available for the Google Web Toolkit [50] development framework. Our widgets were derived from the controls presented earlier and they abstract input from a variety of interaction mechanisms, provide graphical representations, and provide a standard object-oriented programming interface. The following widgets are currently provided.

Button. A button widget represents an action control (select task) and it allows users to trigger actions in the public display application. An action button is graphically represented on the public display as a standard web button with a label (Figure 1(a)).

List Box. The list box widget represents an option list control (select task) and allows users to select among a set of related items. List boxes are graphically represented as a vertical list of text items with a title at the top (Figure 1(e)).

Text Box. A text box widget represents an unbounded text control (data entry task) and it allows users to input free text. Text boxes are graphically represented as standard web text boxes with a label inside (Figure 1(f)).

Upload. An upload widget represents a generic upload control (upload media task) and it allows users to submit media files to the public display application. An upload widget is graphically represented as box with a down arrow and a label inside (Figure 1(c)).

Download. A download widget represents a generic download control (download media task) and it allows the application to provide files that users can download to their personal devices. A download widget is graphically represented as box with an up arrow and a label inside (Figure 1(b)).

Check-in. A check-in widget represents a check-in control (signal presence task) and it allows users to signal the application that they are present. It is graphically represented as a location marker with a label on the side (Figure 1(d)).

These widgets can be interacted with using various mechanisms, but programmers are shielded from the details of the particular mechanism used to interact: all widgets trigger high-level events that are independent of the concrete mechanism used by a particular user. Our toolkit supports the following types of interaction mechanisms. *Text-based interaction* includes various different input mechanisms such as SMS, instant messaging, email, Bluetooth naming, and other mechanisms where the communication is made mainly via text messages. The toolkit generates unique textual references

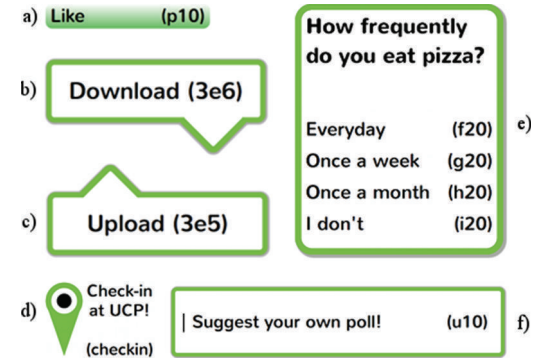

Figure 1: Default graphical representations for widgets.

Figure 2: Widgets in the context of a public display application.

that users input in the text message to allow the system to identify the target application and widget. The toolkit also supports smart-devices by automatically generating a *graphical user interface for mobile devices*. The toolkit is also capable of generating *QR codes* for widgets, allowing interaction with specific application features simply by scanning a visual code. Finally, widgets are also *touch-enabled*, allowing users to interact directly with the application via touch-displays.

We have already created and deployed various interactive public display applications created with this toolkit and we have evaluated it with independent programmers and application users in a real-world setting. Figure 2 shows an example of the Public YouTube Player application created with our toolkit. This application searches for and plays YouTube videos providing several interaction features to users such as "liking" videos that have been recently played; getting the URL of a recently played video to play it in their own devices; selecting a video to be played next from the list of search results; and reporting inappropriate videos. Any user can interact with any of these features at any time, using any of the interaction mechanisms mentioned before.

A full evaluation of this toolkit can be found in [51]. Our experience with the toolkit reinforces the suitability of the design space of interaction controls we have proposed in this paper. We created three applications that had different requirements for interaction features: a video player, a word game, and a polls application. These applications were created without any specific interaction mechanism in mind but they

were deployed and interacted with using different mechanisms (SMS, email, QR codes, and smartphone app). Users successfully interacted with and understood the different types of controls and feedback that were provided by each application. In addition, independent programmers used our toolkit to create interactive content and reported no major difficulty in understanding the concepts behind the provided high-level interaction abstractions that the toolkit provides.

5. Conclusion

We have presented a study about interaction tasks and controls for public display applications, grounded on the existing descriptions of concrete interactive display systems available in scholarly publications. The key contributions of this work are as follows. We have characterized six high-level interaction tasks focused on the specificities of public display interaction, more specifically select, data entry, upload, download, signal presence, and dynamic manipulation. These tasks represent a classification of the major types of interaction between users and public displays; we have also identified various types of concrete interaction controls that may enable those interaction tasks to be integrated into applications for public displays. These controls constitute a first step towards a list of controls that may compose future interaction toolkits for public displays; we have also organized the various interaction mechanism for public displays in a design space adapted from Ballagas et al. [6] that sketches a mapping between the high-level abstractions provided by the interaction tasks that have been identified and the concrete interaction mechanisms that can be implemented by those displays.

We realize that although interaction tasks define different types of information exchanges between user and system, there are borderline cases where different interaction tasks could be used to implement the same interaction feature. For example, an important difference between data entry and select tasks is that in data entry tasks it is generally impossible for the application to enumerate all possible values. In cases where it is possible to enumerate all possible values, the two types of tasks could be used interchangeably. However, using data entry to mimic a select task would require applications to perform extra validation and processing of the received data. This is also valid for other cases, such as uploading, downloading, and signal presence tasks, which could be mimicked by applications using other tasks, at the expense of extra processing and validation.

We also realize that, through abstraction, we lose some of the detail that may be important for certain types of application. For example, in some games, a very-fine grained control of gesturing can be a fundamental part of the playing experience and may not be properly addressed by high-level abstractions. In these cases, the interaction experience is tightly coupled with the interaction mechanism and abstracting the interaction into tasks loses the detail about the bodily movements. For these cases, a different approach would obviously be needed and by no means do we claim with our work to cover the whole interaction design space. Our focus is the broad range of simple interaction techniques that are highly common (essentially the same across different displays systems) and yet currently dependent on totally ad hoc implementation approaches. It is in that space that even small steps towards increased abstraction can make a huge difference towards systems that are more usable and easier to develop.

Finally, we understand that the abstractions embedded in the desktop computing model exist at multiple levels and are the result of many years of evolution in interface design. In this work, we do not aim to reach anywhere near the equivalent of that for public displays, but simply to provide a first step in that direction. With the interaction tasks, the mapping between tasks and mechanisms, and the interaction controls, we have a tool to structure an interaction system for public display applications. This is a valuable tool for allowing application developers to make more informed decisions on the types of controls that they would need, considering for example, the applications goal but also the envisioned interaction modalities. We have made a first demonstration of how this can be achieved through the instantiation of the interactive controls of our design space in the PW toolkit. Hopefully, this design space will be the basis for various others infrastructures, toolkits, and libraries, with different aims and offering different interaction models, contributing to open up the development of interactive public display applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

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

References

[1] A. Bellucci, A. Malizia, P. Diaz, and I. Aedo, "Human-display interaction technology: emerging remote interfaces for pervasive display environments," *IEEE Pervasive Computing*, vol. 9, no. 2, pp. 72–76, 2010.

[2] N. Davies, M. Langheinrich, R. Jose, and A. Schmidt, "Open display networks: a communications medium for the 21st century," *Computer*, vol. 45, no. 5, pp. 58–64, 2012.

[3] R. R. Swick and M. S. Ackerman, "The X toolkit: more bricks for building user interfaces, or widgets for hire," in *Proceedings of the Usenix Winter 1988 Conference*, pp. 221–228, 1988.

[4] J. Mackinlay, S. K. Card, and G. G. Robertson, "Semantic analysis of the design space of input devices," *Human-Computer Interaction*, vol. 5, no. 2-3, pp. 145–190, 1990.

[5] J. D. Foley, V. L. Wallace, and P. Chan, "The human factors of computer graphics interaction techniques," *IEEE Computer Graphics and Applications*, vol. 4, no. 11, pp. 13–48, 1984.

[6] R. Ballagas, M. Rohs, J. G. Sheridan, and J. Borchers, "The Design Space of Ubiquitous Mobile Input," in *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*, J. Lumsden, Ed., vol. 1, pp. 386–407, IGI Global, 2008.

[7] G. F. P. Deecker and J. P. Penny, "Standard input forms for interactive computer graphics," *ACM SIGGRAPH Computer Graphics*, vol. 11, no. 1, pp. 32–40, 1977.

[8] B. A. Myers, "New model for handling input," *ACM Transactions on Information Systems*, vol. 8, no. 3, pp. 289–320, 1990.

[9] L. Bass and J. Coutaz, *Developing Software for the User Interface*, Addison Wesley, 1991.

[10] D. A. Norman, *The Design of Everyday Things*, Basic Books, 2002.

[11] F. Paterno, C. Santoro, and L. D. Spano, "MARIA: a universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments," *ACM Transactions on Computer-Human Interaction*, vol. 16, no. 4, article 19, pp. 1–30, 2009.

[12] L. Terrenghi, A. Quigley, and A. Dix, "A taxonomy for and analysis of multi-person-display ecosystems," *Personal and Ubiquitous Computing*, vol. 13, no. 8, pp. 583–598, 2009.

[13] A. J. Dix and C. Sas, "Mobile personal devices meet situated public displays?: synergies and opportunities," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 1, no. 1, pp. 11–28, 2010.

[14] R. Ballagas, M. Rohs, and J. G. Sheridan, "Sweep and point and shoot: phonecam-based interactions for large public displays," in *Proceedings of the CHI conference on Human factors in computing systems*, pp. 1200–1203, 2005.

[15] R. José, N. Otero, S. Izadi, and R. Harper, "Instant places: using bluetooth for situated interaction in public displays," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 52–57, 2008.

[16] N. Davies, A. Friday, P. Newman, S. Rutlidge, and O. Storz, "Using bluetooth device names to support interaction in smart environments," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, pp. 151–164, June 2009.

[17] K. Cheverst, A. Dix, D. Fitton et al., "Exploring bluetooth based mobile phone interaction with the hermes photo display," in *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '05)*, pp. 47–54, September 2005.

[18] D. Dearman and K. N. Truong, "BlueTone: a framework for interacting with public displays using dual-tone multi-frequency through bluetooth," in *Proceedings of the 11th ACM International Conference on Ubiquitous Computing (UbiComp '09)*, pp. 97–100, October 2009.

[19] LocaModa, "LocaModa App Store," 2010, http://locamoda.com/.

[20] T. Paek, M. Agrawala, S. Basu et al., "Toward universal mobile interaction for shared displays," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '04)*, pp. 266–269, usa, November 2004.

[21] D. W. McDonald, J. F. McCarthy, S. Soroczak, D. H. Nguyen, and A. M. Rashid, "Proactive displays: supporting awareness in fluid social environments," *ACM Transactions on Computer-Human Interaction*, vol. 14, no. 4, article 16, pp. 1–30, 2008.

[22] D. Vogel and R. Balakrishnan, "Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users," in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, pp. 137–146, October 2004.

[23] N. Sawhney, S. Wheeler, and C. Schmandt, "Aware community portals: shared information appliances for transitional spaces," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 66–70, 2001.

[24] T. Ojala, V. Kostakos, H. Kukka et al., "Multipurpose interactive public displays in the wild: three years later," *Computer*, vol. 45, no. 5, pp. 42–49, 2012.

[25] B. Glaser and A. Strauss, *The Discovery of Grounded Theory*, Aldine Transaction, 1967.

[26] J. C. S. Cardoso and R. José, "Interaction tasks and controls for public display applications—list of analysed publications," *Figshare*, 2014.

[27] A. Cooper, R. Reimann, and D. Cronin, *About Face 3: The Essentials of Interaction Design*, John Wiley and Sons, New York, NY, USA, 2007.

[28] E. Toye, R. Sharp, A. Madhavapeddy, and D. Scott, "Using smart phones to access site-specific services," *IEEE Pervasive Computing*, vol. 4, no. 2, pp. 60–66, 2005.

[29] E. F. Churchill, L. Nelson, L. Denoue, J. Helfman, and P. Murphy, "Sharing multimedia content with interactive public displays: A case study," in *Proceedings of the conference on Designing Interactive Systems: Across the Spectrum (Dis '04)*, pp. 7–16, August 2004.

[30] J. F. McCarthy, S. D. Farnham, Y. Patel et al., "Supporting community in third places with situated social software," in *Proceedings of the 4th international conference on Communities and technologies (C&T '09)*, pp. 225–234, 2009.

[31] F. Alt, T. Kubitza, D. Bial et al., "Digifieds: insights into deploying digital public notice areas in the wild," in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia (MUM '11)*, pp. 165–174, 2011.

[32] M. Rohs, "Visual code widgets for marker-based interaction," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops*, pp. 506–513, 2005.

[33] A. Grasso, M. Muehlenbrock, F. Roulland, and D. Snowdon, "Supporting communities of practice with large screen displays," in *Public and Situated Displays: Social and Interactional Aspects of Shared Display Technologies*, K. O'Hara, E. Perry, E. Churchill, and D. M. Russel, Eds., pp. 261–282, Kluwer Academic, 2003.

[34] K. Martin, A. Penn, and L. Gavin, "Engaging with a situated display via picture messaging," in *Proceedings of the CHI conference on Human factors in computing systems (CHI '06)*, 2006.

[35] S. Carter, E. F. Churchill, L. Denoue, J. Helfman, and L. Nelson, "Digital graffiti: public annotation of multimedia content," in *Proceedings of the CHI conference on Human factors in computing systems (CHI '04)*, pp. 1207–1210, 2004.

[36] H. Brignull, S. Izadi, G. Fitzpatrick, Y. Rogers, and T. Rodden, "The introduction of a shared interactive surface into a communal space," in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '04)*, pp. 49–58, November 2004.

[37] A. Ferscha, G. Kathan, and S. Vogl, "WebWall: an architecture for public display WWW services," in *Proceedings of the International World Wide Web Conference*, 2002.

[38] H. Raj, R. Gossweiler, and D. Milojicic, "ContentCascade incremental content exchange between public displays and personal devices," in *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS '04)*, pp. 374–381, August 2004.

[39] R. Hardy and E. Rukzio, "Touch & interact: touch-based interaction of mobile phones with displays," in *Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '08)*, pp. 245–254, September 2008.

[40] D. M. Russell and R. Gossweiler, "On the design of personal and communal large information scale appliances," in *Proceedings of the 3rd international conference on Ubiquitous Computing (UbiComp '01)*, pp. 354–361, 2001.

[41] R. José, H. Pinto, B. Silva, A. Melro, and H. Rodrigues, "Beyond interaction: tools and practices for situated publication in display networks," in *Proceedings of the International Symposium on Pervasive Displays (PerDis '12)*, pp. 1–6, 2012.

[42] B. Shneiderman, "Direct manipulation: a step beyond programming languages," *Computer*, vol. 16, no. 8, pp. 57–69, 1983.

[43] P. Peltonen, E. Kurvinen, A. Salovaara et al., "'It's mine, don't touch!': interactions at a large multi-touch display in a city centre," in *Proceeding of the 26th Annual CHI Conference on Human Factors in Computing Systems (CHI '08)*, pp. 1285–1294, April 2008.

[44] Ydreams, "Vodafone Cube," 2003, http://www.ydreams.com/#/en/projects/publicurbanexperiences/giantinteractivebillboards-vodafone/.

[45] B. A. Myers, H. Stiel, and R. Gargiulo, "Collaboration using multiple PDAs connected to a PC," in *Proceedings of the 7th ACM Conference on Computer Supported Cooperative Work (CSCW '98)*, pp. 285–294, November 1998.

[46] J. Müller, R. Walter, G. Bailly, M. Nischt, and F. Alt, "Looking glass: a field study on noticing interactivity of a shop window," in *Proceedings of the ACM annual conference on Human Factors in Computing Systems (CHI '12)*, 2012.

[47] T. Ojala, H. Kukka, T. Lindén et al., "UBI-hotspot 1.0: large-scale long-term deployment of interactive public displays in a city center," in *Proceedings of the 5th International Conference on Internet and Web Applications and Services (ICIW '10)*, pp. 285–294, May 2010.

[48] A. Bragdon, R. DeLine, K. Hinckley, and M. R. Morris, "Code space: Touch + Air gesture hybrid interactions for supporting developer meetings," in *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS '11)*, pp. 212–221, November 2011.

[49] J. C. S. Cardoso and R. José, "PuReWidgets: a programming toolkit for interactive public display applications," in *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems (EICS '12)*, pp. 51–60, 2012.

[50] Google, "Google Web Toolkit," 2011, http://code.google.com/webtoolkit/.

[51] J. C. S. Cardoso and R. José, "Evaluation of a programming toolkit for interactive public display applications," in *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia (MUM '13)*, pp. 1–10, 2013.