Tactive, a Framework for Cross Platform Development of Tabletop Applications

Ombretta Gaggi[1] and Marco Regazzo[2]
[1]*Department of Mathematics, University of Padua, via Trieste, 63, Padova, Italy*
[2]*Anyt1me S.r.L., via Siemens 19, Bolzano, Italy*

Keywords: Tabletop Applications, Touch Interfaces, Webkit Engine.

Abstract: The number and types of applications developed for multi-touch tabletops are dramatically increased in the last years, mainly due to the fact that interactive tabletops allow a more natural interaction with the user through their multi-touch interfaces. Despite many applications share a big set of common features, e.g., gestures recognition, interface orientation, etc., almost all applications implement their home made software solutions. In this paper we present *Tactive*, a software layer for fast development of *portable* applications for multi-touch interactive tabletops. *Tactive* allows to abstract from hardware and software equipment and to embed a web application into a application for multi-touch surfaces. Our framework supports up to five fingers gestures recognition and communication between different windows, and allows to save more than 60% of developing time.

1 INTRODUCTION

In the last few years, the market of multi-touch tables is experiencing a situation very similar to what happens in the mobile applications market. The number of developed applications is dramatically increased, interactive tabletop surfaces are used to improve learning activities (Rick et al., 2011), inside museums (Geller, 2006), where the diversity of visitors create a natural laboratory for testing this kind of interface, to help the management of emergency (Qin et al., 2012), and in many other collaborative activities like, e. g., photoware (Pedrosa et al., 2013), etc. Consequently, also the the number of hardware solutions increased, each one requiring a particular SDK, programming language, etc.

Like smartphones, interactive tabletops allow a more natural interaction with the user through their multi-touch interfaces (Forlines et al., 2007) and, unlike mobile devices, they allow the interaction of more than one user at the same time. Tabletops promote collaboration and social experiences, and can act as a meeting point.

The possibility to interact with multiple user at the same time requires an important set of new features for the user interface: e. g. the user are placed around the table, therefore they need different orientations of the interaction widgets, they can collaborate using the same space and objects or can compete for them. Therefore all applications developed for this kind of interface have to face a common set of problems, e. g., the recognition and management of particular gestures and the orientation of the interactive widgets. Despite these common features, almost all the applications implement all these features starting from scratch, since no software solution exists.

In this paper we present our tabletops solution, which can be applied to different size of surface (42" or more) and can work upon different operating system. We have developed a software layer which is able to abstract from hardware and software constraints of the device in which it is installed (screen size, operating system, etc) and allows the developer to easily manage common features of the user interface discussed above. Our solution provide a Javascript API which allows a developer to build an entire tabletop application only using web technologies, in particular HTML5, CSS3 and Javascript, without the need to know anything about the underlying hardware and software.

Moreover, our solution allows a very easy reuse of web applications already developed for non touch interfaces, and personalization of the final application.

Finally, we provide our multi-touch tabletop with a strong interior design (see Figure 1) on its shape and materials while most of the hardware available on the

Gaggi O. and Regazzo M..
Tactive, a Framework for Cross Platform Development of Tabletop Applications.
DOI: 10.5220/0004857800910098
In *Proceedings of the 10th International Conference on Web Information Systems and Technologies* (WEBIST-2014), pages 91-98
ISBN: 978-989-758-024-6

Figure 1: Our tabletop solution was designed by an expert interior architect.

market is little more than a "big iPad mounted on four legs".

The paper is organized as follows: Section 2 discusses the related works and the need for a framework for the development of portable multi-touch applications. Section 3 presents *Tactive*, a software layer which provides a set of features and gestures to speed up the design process of multi-touch interactive applications. A set of success stories about applications developed with the framework is discussed in Section 4. Finally, we conclude in Section 5.

2 RELATED WORKS AND BACKGROUND

There are a lot of applications for interactive tabletops and surfaces described in literature. Correia et al (Correia et al., 2010) described an application for museum setting. A tabletop is used to enhance user experience presenting semantic information about all the artworks of the exhibition. The authors realized a tabletop setup based on the Frustrated Total Internal Reflection system. More than one user can interact with the tabletop at the same time in a collaborative way. The user interface is an ad-hoc application, build from scratch with the help of some open framework.

uEmergency (Qin et al., 2012) is a emergency management system based on a very large multi-touch surface. The users can collaborate in a very intuitive way around a table which displays available information on the ongoing emergency. It allows people to carry out face-to-face communication based on a horizontal map. Users can also analyzed real-time situation with fingers or digital pens. A study shows that the use of interactive surfaces improves efficiency in decision making and collaboration for coping with an emergency.

Pedrosa et al, used tabletops to explore home videos and photos (Pedrosa et al., 2013). A set of 24 users evaluates an application which displays photos and videos on a horizontal touch surface to allow storytelling and random exploration. The authors show that, among collaborative tools also personal spaces within the tabletop were useful for allowing independent navigation.

The application described so far have many common features: they are highly visual systems, mainly controlled by touches and some common gestures performed on the surface of the system, e. g., browsing a collection of items, selecting a particular item, accessing a documents, and so on. All these applications can interact with several users at the same time, and each user requires a different orientation of the interface, according to his/her position. Despite many common requirements, the developers of all these applications need to implement the majority of these features from scratch and the available frameworks provide only very low level features.

As a general remark, designer of multi-touch applications do not have a reference model to model user interface and interaction, but often rely on best practice and intuition rather than on a systematic development process (Wigdor et al., 2009). For this reason, many works in literature address the problem of designing user interface for interactive surfaces (Anthony et al., 2012; Hesselmann et al., 2011; Luyten et al., 2010; Nielsen et al., 2004; Seto et al., 2012; Urakami, 2012).

Urakami (Urakami, 2012) has shown that the user choice of gestures was affected by the size of the manipulated object, expertise, and nature of the command (direct manipulation of objects vs. assessment of abstract functions), therefore it is essential to involve the user in the development of gesture vocabularies. The same approach is followed by Hesselmann et al, that proposed an iterative process of five steps tailored to the development of interactive tabletops and surfaces applications, called SCiVA, Surface Computing for Interactive Visual Applications. The key idea of SCiVA is to strongly involve the user in the design process to improve the usability of the final product (Hesselmann et al., 2011).

Luyten et al, try to reach a consensus on a set of design patterns that aid in the engineering of multi-touch interfaces and transcend the differences in available platforms (Luyten et al., 2010). Seto et al, investigate the problem of how to manage menus displacement in multi-user surfaces(Seto et al., 2012). In particular they focus on the discoverability of system menus on digital tabletops designed for public settings. This study presents a set of design recommendations to improve menu accessibility: e. g., discernible and recognizable interface elements, such as buttons, supported by the use of animation, can effec-

tively attract and guide the discovery of menus.

This analysis of the literature shows that some steps toward the definition of design patterns for the development of interactive multi-touch interfaces have been done, but there are not already built *off-the-shelf* components to create these interfaces, but each application build from scratch its user interface. Native frameworks, like Microsoft Surface 2.0 SDK and Runtime (Microsoft, 2013a), Windows Presentation Foundation + Native Touch recognition by Microsoft Windows 8 (Microsoft, 2013b) and Smart Table SDK (SMART Technologies, 2013), help to develop multi-touch applications but require a particular hardware/software configuration.

Our goal is the creation of a software layer, portable on each operating system and hardware solution, which provides this set of features and gestures to speed up the design process of multi-touch interactive applications by avoiding to re-invent the wheel each time (Gaggi and Regazzo, 2013).

Other solutions exist which addresses a similar problem. Glassomium (Toffanin, 2013) is a project based on web technologies which aims to port web applications to multi-touch surface. Even if the key idea is quite the same, it allows for rotations, scaling and dragging even through an unstable beta and it is not able to identify gestures which involve the whole hand, Glassomium can be considered a windows manager, which allows to recognize the user gestures and to manage them, but it does not implements cross-windows communication, therefore, it lacks of a proper mechanism to change the user experience on the base of the interaction of other users. To the best of our knowledge this feature is implemented only by our solution.

GestureWorks (Ideum, 2013) and Arena (Unedged, 2013) are frameworks which provide generic and cross platform functionalities, like gestures recognition, to develop touch applications, but they are not able to manage more than one application being launched at the same time or multiple application enclosed in different windows.

3 DESCRIPTION OF THE FRAMEWORK

In this section we discuss the design issue and the implementation details of the developed software layer, called *Tactive*. *Tactive* is a framework, which allows to speed up the development of applications for multi-touch surfaces. This goal is reached since:

- *Tactive* provides a way to encapsulate web applications into widgets suitable for multi-touch surfaces, therefore already developed web applications can be easily adapted to multi-touch interactive surfaces;
- *Tactive* allows to abstract from hw/sw details: an entire application can be developed using web technologies, therefore we do not ask the developer to know any particular language or technology bound to the particular hw/sw equipment, he or she only needs to know how to use the Javascript API provided by our framework;
- applications developed with *Tactive* are able to adapt themselves to different size of the surface (*Tactive* helps to realize the so-called *fluid* applications) and
- *Tactive* provides a set of features common to multi-touch applications like windows disposition, gestures recognition and interface orientation.

3.1 System Architecture

The architecture of our system is depicted in Figure 2. *Tactive* is organized in two levels. The lower one, called the *O.S. Layer* guarantees the independence from the underlying hardware: it contains the operating system (MS Windows 7, MS Windows 8 and Linux are supported), and a set of protocol and libraries to manage touch gestures if the chosen operating system does not support them natively. The *Application Layer* manages the applications, their windows and the interaction between the applications and the user or between different applications.

Tactive clearly separates the contents, displayed to the users, from the interaction widgets and the software components used to display the contents. For this reason, the architecture of our framework contains a content manager, called *Application Container*, which manages how to display the contents, and a windows manager.

The Application Container allows the division between contents and interaction widget using *WebViews*, i. e., components that display web pages. A WebView allows to embed HTML pages inside an application. This component uses the *WebKit* rendering engine to display web pages inside a window of the application, and includes methods to navigate forward and backward through a history, to zoom in and out, to perform text searches, etc.

The Application Container is the underlying component that encapsulate all the functionalities needed to interact with the user and with other components within the table, i. e., the *Touch Manager* that allows gestures management and recognition, and the

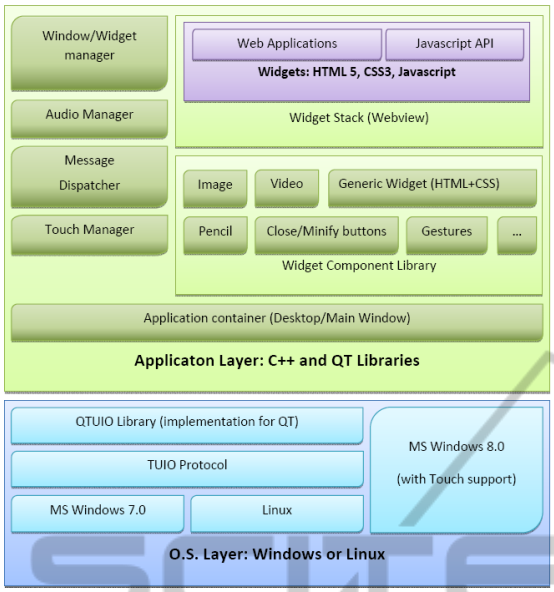

Figure 2: Architecture of the developed software layer for *Tactive*.

Window/Widget Manager that provides the stack of visible objects (see Figure 2). It is also responsible to collect and enumerate application specific contents (e. g., images, videos, web pages or multimedia items) that are stored as web pages and rendered through the WebView.

The frameworks supports both *on-line* and *off-line* content/pages but usually the second option (a local web server) is preferred to let the application works and displays contents even in absence of an Internet connection.

Widgets for the visualization of media items like videos and images have been implemented using *WebViews*. *Tactive* has been designed to be extendible: an expert developer may create a new component extending the widget component (or one of its subclasses), automatically taking advantage of all the features already implemented and described above[1].

Using our framework, content can be created by a web developer (that designs the structure) and update by a content editor.

The mechanism of *WebViews* is used to developed hybrid applications for mobile devices, i. e., applications based on the HTML5 languages which are wrapped with a webkit engine and rendered as native mobile applications. PhoneGap (Apache Software Foundation, 2013), also known as Apache Cordova, is a framework for cross-platform mobile development which create hybrid applications. Our approach is very similar: the idea is to take advantage of the portability of web technologies to develop portability of applications for multi-touch interactive surfaces.

Using a WebView, the developer only need to specify which is the web page to render. Therefore contents has to be enclosed into web pages to be displayed to users. At this point, our framework allows the visualization of contents into a window on the tabletop.

Contents can be arranged (and personalized, e. g. using a particular layout) using the CSS standard language like what happens for web sites. But the provided interaction is very poor, since the user can touch the interface, but the touch is interpreted like a movement of a mouse pointer. No gestures like pinch, rotation or drag are supported, but only tap and double tap.

Since people do not use their hand and fingers like a mouse pointer, we need the *Touch Manager* component to manages concurrent touches and gestures of many users. This software component manages portability of touches and gestures recognition and implements the TUIO protocol (Kaltenbrunner et al., 2013) which allows the transmission of an abstract description of interactive surfaces, including touch events and tangible object states. This protocol encodes control data from a tracker application (e.g. based on computer vision) and sends it to any client application that is capable to decode the protocol. Technically TUIO is based on Open Sound Control (OSC) - an emerging standard for interactive environments not only limited to musical instrument control - and can be therefore easily implemented on any platform that supports OSC.

The recognition of the gestures is managed extending qTUIO (Belleh and Blankenburgs, 2013), a library which implements a TUIO listener on a local UDP socket and forwards the events into the internal event system of Qt. qTUIO is able to recognize gestures, e. g., dragging of an object, made with one finger, two fingers are allowed only for the zoom in and out management. Since the user usually move windows and objects with the whole hand, qTUIO is only a first step through the realization of a portable software for the complete management of multi-touch interaction. For this reason, the Touch Manager extends this library to recognize and manage also gestures which involves more than one finger, e. g., multi-touch pan and pinch, scroll, drag and rotation using up to five fingers.

Since *Tactive* allows to launch more than one applications at the same time, another problem arise,

[1]We must note here that the framework development is almost complete, therefore, even if *Tactive* is extendible, it is very difficult that a developer of applications needs to implement a new type of widget.

i. e., the management of application audio. In fact, if many applications use contemporary the audio interface, the result can be a big uproar, and it could be very difficult for the users to understand the audio messages. Consider, as an example, the case in which two users play contemporary two demonstrative videos, what happens is that the audio messages are overlapped and none of the users is able to easily follow the video. The situation is even worse when dealing with more users.

For this reason, *Tactive* implements the component called *Audio Manager*, which is able to manage contemporary audio. Audio messages are classify by the content editor according to their nature, i. e. soundtrack or spoken comment. More soundtracks can plays together, two spoken comments cannot, so one of the two audio (and video if it is the audio comment of a video) is suspended till the end of the first one. To decide which audio is paused, the Audio Manager allows to define priority classes, or use a first-in, first-served policy if no priority was defined by the content editor.

3.2 Communication between Different Windows

An important component of our architecture is the *Windows Manager*. Given the dimension of the tabletop, concurrent interactions by more than one users is an important issue to consider. As an example, the users can compete for space on the surface. For these reason, when a new window is opened (even by a new user or not), this operation can require the resize of all the other windows already present on the table. Otherwise, actions from a particular user may affect the behavior of the windows of other users. To allow the easily implementation of applications with this kind of features, *Tactive* implements a windows manager and communication protocol between windows provided by the *Message Dispatcher*.

Let us consider as example, an application with a map, e. g., a map of a city with the list of its museums, or a map of an exhibition with the position of the stands. The map can be rendered with HTML5 on a WebView (see Figure 3). If the user touches a museum or a stand the application opens a new window, with the web site of (or a page dedicated to) the museum/stand, and the user can interact with this window, resize it, or move across the table. If the user touches the "go to the map" button on the new windows, the initial window with the map is moved over the current window of the user. Figure 3 shows a screenshot from an application developed for a local fair.

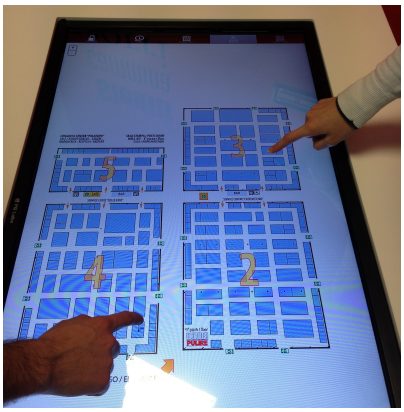

Figure 3: Screenshot from an application developed for a local exhibition.

To implement this behavior, a communication protocol between windows has been developed. The communication protocol allows the developer to change the content or the behavior of a window on the base of the behavior, or user interaction with, another windows. Each *WebView* communicates with the software layer *Tactive*, which acts as a windows manager. We need a windows manager instead of a simple communication protocol between windows, widgets or WebViews because only the windows manager knows how many windows are currently open in the surface, where they are, and how they are interacting with the user, each window knows only the information about itself, and nothing about the other. Moreover, the use of a windows manager allows an easy recover from the failure of a single window, since the manager records a set of information for each window and is able to stop, suspend or restart it.

Tactive implements the windows management using the C++ language to address performance issues. Moreover, it offers to developers of multi-touch application a Javascript API to manage events triggered by *Tactive* inside their web applications which use our framework to work on multi-touch interactive surfaces. The Javascript API allows to enlarge, resize, minimize, close or move a window, in response to a user interaction, also on other windows.

Moreover, using this API, it is possible to send a message to a widget active on another window through the Message Dispatcher. Consider as example an *advergame*: the user gains coins to play with a slot machine, answering to a questionnaire. When he/she completes the questionnaire, the window with the questions sends a message to the slot machine, enabling the user to play. This communication between windows is enabled by the Javascript API, which is used to compose the message and trigger the event through the Message Dispatcher to the

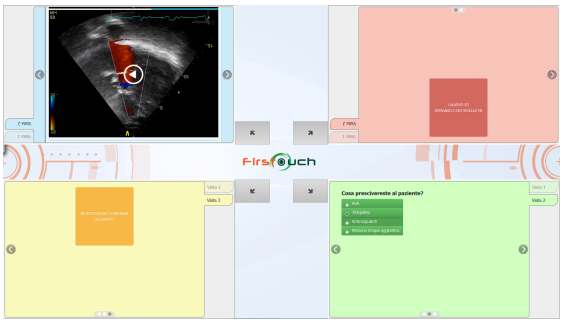

Figure 4: Screenshot from an application developed for doctors training.

Windows Manager, which is in charge of triggering the right response to the right window.

4 CASE STUDIES AND DISCUSSION

The framework *Tactive* has been used to develop six applications in completely different contexts, ranging from fair exposition to the launch of a new product. In this section we describe two success stories and we report some data about how the use of this framework deeply impacts the development of a multi-touch application.

The first success story is an application to improve learning activities developed for a local company. The context of use was the training of physicians. The application puts around a table four physicians, two per side. Each physicians has different materials and documents, i. e., medical records, laboratory diagnosis, x-rays, etc, about a single patience with a particular disease. No physicians has enough material to understand which is the disease which affects the patient without the help of data held by other doctors.

Figure 4 shows a screenshot of the interface. Different content is delivered to each workspace. The goal is to improve communication strategies and the ability to work together of the physicians. The doctors can create new windows to share the content, can drag the window around the table surface, rotate, zoom in and out to better understand a picture, e.g., an x-ray, or a video, e. g., an ultrasound scan. When a doctor puts in common his own material dragging it on the center of the table, the other windows are minimized, to better focus the other doctors' attention on that particular medical data.

The application was created using *Tactive*, therefore the developer only needs to assemble the content into web pages. The Javascript API was used to implement the communication between windows, i. e., to minimize all the windows when a physician puts some data on the center of the table.

Figure 5: Screenshot from an application developed for car market.

Thanks to our software layer, the development process is reduced to content creation which requires 45 man-days of a developer for its realization. The development of the same application using the C++ language on a TouchWindow Slice Table Multi-Touch (Touchwindow S.r.l., 2013) required one man-year, therefore our framework allows to save about 86% of time [2] as reported in Table 1.

The same application was used during 15 different one-day courses for physicians, using the same structure, and changing only the content, i. e., the text in the web pages, but not the structure of the pages. This adaptation process required only one day of work of a web content editor. Moreover, the application can run on any tabletop, independently from the operating system[3] or the size of the surface.

The second case study is an application developed for the launch of a new product of a leading company in the car market. In this case, the application was used by a single speaker who, during his presentation, switched between an interactive slideshow, several videos and some online demos on a web site. Figure 5 shows the menu which allows to choose a video for the presentation.

The main issue for this application was to mix both off-line and online content: the "traditional" software building blocks used for tabletop UI would have required to develop the application from scratch, loading it with the off-line content (videos and slideshows) and linking online content into a webview or a browser. Such application would have required four weeks of a FTE (Full Time Equivalent) software developer, and an implementation using Flash would have required ten man-days, as reported in Table 1.

[2]This information has been extracted from a previous realization of the same application, which was not independent from the chosen hardware.

[3]Microsoft Windows 7 or superior, Linux and Apple iOs Lion are supported.

Table 1: Impact of the use of *Tactive* in the developing time for the 6 application developed using this framework. The developing time is expressed in man-days. We suppose that a man-week is equal to 5 man-days, a man-month correspond, on average, to 20 man-days, and finally, a man-year corresponds, on average, to 220 man-days.

App	Tactive	Flash	Saving	C++	Saving
Success Story 1	30	–	–	220	86%
Success Story 2	3	10	70%	20	85%
Sculptor Exhibition	3	10	70%	15	80%
Innovation Festival	5	20	75%	60	91%
Job Event	2	5	60%	20	–
Learning App	2	5	60%	20	–

Using the *Anytable* framework, any piece of content was linked into a different web page and published online, included the main menu page: the overall activity required 3 days of a FTE web developer, therefore it saves 85% of time with respect to an implementation using a native SDK, and 70% of time with respect to Flash implementation.

The framework has been used to implement other four applications, for a sculpture exhibition, two fairs and another type of application for learning with a different interaction with the users. Table 1 reports the required time to implement these applications using our framework. These results are compared with the estimated time required to develop the same applications using Flash and C++ language with a native SDK solution. This information has been collected from quotations that have been made during the sale phases of the final product to the customer.

We can see that our framework allows to save between 60% and 75% of developing time respect to Flash implementation. This important range of percentage rises to 80% and 91% for applications developed using C++ language and a native SDK. It is easy to note that this saving is higher for complex applications.

Although this important result in terms of time saving, our framework introduces also some drawbacks. In particular, to allow independence from the underlining hardware, we abstract from its characteristic and we implement a software layer which is able to operate with any tabletop. This means that *Tactive* defines a set of functions common to all tabletop solutions, and does not consider features which are available only on a particular hardware configuration: this choice limits the expressiveness of *Tactive*, which does not allow to use manufacturer-specific features in applications development. However, further development of HTML5 API will be considered in the future release of our software in order to lower this limitation.

5 CONCLUSION

In this paper we present *Tactive*, a software layer for fast development of *portable* applications for multi-touch interactive tabletops. The framework is based on modern web technologies and its core unit is developed using the C++ language.

The novelty of our approach consists in three points:

- the development of a framework for the creation of application for multi-touch surfaces which are independent from the hardware and software equipment;
- the possibility to use (and possible re-use) web pages decreases the time spent to develop the multi-touch applications and does not require to learn any new technology. Our experiments shows that *Tactive* allows an important reduction in time needed for development, between 60% and 91%;
- finally, no other software framework provides an easy communication between different windows of the same applications.

Moreover, our framework extends the qTUIO library to manage the recognition of gestures made with up to five fingers.

Future works will be dedicated to the implementation of an API to manage Near Field Communication (NFC). NFC is a technology that provides short-range (up to a maximum of 10 cm) and bi-directional wireless connectivity. The idea is to save the state of the user, in term of opened documents and windows, and which is the window currently active, and to re-create the entire workspace at the correct state, every time that user approaches the system.

REFERENCES

Anthony, L., Brown, Q., Nias, J., Tate, B., and Mohan, S. (2012). Interaction and recognition challenges in interpreting children's touch and gesture input on mo-

bile devices. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ITS '12, pages 225–234.

Apache Software Foundation (2013). Phonegap, http://phonegap.com/.

Belleh, W. and Blankenburgs, M. (2013). qTUIO Library. http://qtuio.sirbabyface.net/.

Correia, N., Mota, T., Nóbrega, R., Silva, L., and Almeida, A. (2010). A multi-touch tabletop for robust multimedia interaction in museums. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 117–120.

Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 647–656.

Gaggi, O. and Regazzo, M. (2013). An environment for fast development of tabletop applications. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 413–416.

Geller, T. (2006). Interactive tabletop exhibits in museums and galleries. *IEEE Comput. Graph. Appl.*, 26(5):6–11.

Hesselmann, T., Boll, S., and Heuten, W. (2011). Sciva: designing applications for surface computers. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering interactive computing systems*, EICS '11, pages 191–196.

Ideum (2013). Gestureworks Core. http://gestureworks.com/pages/core-home.

Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. (2013). TUIO Framework. http://www.tuio.org/.

Luyten, K., Vanacken, D., Weiss, M., Borchers, J., Izadi, S., and Wigdor, D. (2010). Engineering patterns for multi-touch interfaces. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '10, pages 365–366.

Microsoft (2013a). Surface 2.0 SDK. http://msdn.microsoft.com/en-us/library/ff 727815.aspx.

Microsoft (2013b). Walkthrough: Creating Your First Touch Application. http://msdn.microsoft.com/en-us/library/ee649090.aspx.

Nielsen, M., Strring, M., Moeslund, T., and Granum, E. (2004). A procedure for developing intuitive and ergonomic gesture interfaces for hci. In Camurri, A. and Volpe, G., editors, *Gesture-Based Communication in Human-Computer Interaction*, volume 2915 of *Lecture Notes in Computer Science*, pages 409–420. Springer Berlin Heidelberg.

Pedrosa, D., Guimarães, R. L., da Graça Pimentel, M., Bulterman, D. C. A., and Cesar, P. (2013). Interactive coffee table for exploration of personal photos and videos. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 967–974.

Qin, Y., Liu, J., Wu, C., and Shi, Y. (2012). uEmergency: a collaborative system for emergency management on very large tabletop. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ITS '12, pages 399–402.

Rick, J., Marshall, P., and Yuill, N. (2011). Beyond one-size-fits-all: how interactive tabletops support collaborative learning. In *Proceedings of the 10th International Conference on Interaction Design and Children*, IDC '11, pages 109–117.

Seto, M., Scott, S., and Hancock, M. (2012). Investigating menu discoverability on a digital tabletop in a public setting. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ITS '12, pages 71–80.

SMART Technologies (2013). SMART Table SDK. http://downloads01.smarttech.com/media/products/sdk/smart-table-sdk-summary.pdf.

Toffanin, P. (2013). Glassomium Project. http://www.glassomium.org/.

Touchwindow S.r.l. (2013). Slice Table Multi-Touch http://www.touchwindow.it/en/slice-multi-touch-table.php/.

Unedged (2013). Arena Multitouch Platform. http://arena.unedged.com/.

Urakami, J. (2012). Developing and testing a human-based gesture vocabulary for tabletop systems. *Human Factors*, 54(4):636–653.

Wigdor, D., Fletcher, J., and Morrison, G. (2009). Designing user interfaces for multi-touch and gesture devices. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2755–2758.