# Continuous Tilting Interaction Techniques on Mobile Devices for Controlling Public Displays

**Linda Di Geronimo, Andrea Canonica, Maria Husmann, Moira C. Norrie**
Department of Computer Science, ETH Zurich
{lindad, husmann, norrie}@inf.ethz.ch, canandre@student.ethz.ch

## ABSTRACT
The use of mobile devices to interact with public or semi-public displays has been widely studied. Researchers have investigated the potential use of motion gestures in such settings to allow users to control the display without having to shift either their gaze from the display or their hand position on the device. However, tilting gestures still offer several implementation challenges and multiple variants of such gestures have been proposed. For this reason, we designed a framework that supports the rapid development of web-based applications and enables experimentation with a variety of techniques for continuous tilting interactions. Moreover, we present a study of three different motion gestures (*Constant Move*, *Balance Board* and *Mapped Container*) that can be used to control the cursor on a remote screen.

### ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI):: Interaction styles, User-centered design

### Author Keywords
public semi-public screens, motion Gestures, frameworks

## INTRODUCTION
Nowadays, public displays are found in a myriad of locations, often advertising products or services in streets, airports and bus or train stations. Within organisations, public and semi-public displays are now used to raise community awareness as well as provide information about news and events.

However, support for user interaction remains an issue in terms of both the functionality and modes that could or should be supported. A lot of research has focused on the use of smartphones for interacting with large displays both at home and in public spaces [9, 13]. Most of this research relies on touch gestures, although some work has investigated the alternative of performing motion gestures with a smartphone to interact with a large display [2, 8].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
*EICS '17*, June 26–29, 2017, Lisbon, Portugal
©ACM. ISBN 978-1-4503-5083-9/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3102113.3102120

A potential advantage of tilting interactions over touch in such settings is the fact that these gestures can often be performed by the user without shifting their gaze from the large display or changing their hold position on the phone [8]. For example, continuous tilting interactions allow users to remotely control a cursor on the bigger monitor by continuously moving their device, without requiring them to look at the smaller screen.

Researchers have proposed a number of alternative types of continuous tilting interactions [18, 14], each possibly combined with touch gestures, tactile or visual feedback, and potentially influenced by a multitude of parameters. Consequently, motion interactions still require a lot of design and implementation effort. Our aim therefore was to investigate how we could better support developers as well as carrying further detailed studies on the use of such interactions in cross-device applications.

In this paper, we present two key contributions. First, the WebGravity framework, now publicly accessible[1], offers researchers and developers the opportunity to easily experiment with different types of motion gestures on a variety of devices. Second, to demonstrate the framework's capability, we implemented a set of tilting interactions previously studied in research [2, 18, 14] and applied them in the particular cross-device setting of using a smartphone to interact with a public display. We discuss our findings and provide some insights into our overall experience of using motion gestures.

## BACKGROUND
While the purpose, technology and physical positioning of public and semi-public displays vary, most do not offer any form of interactivity to the user. They can thus be considered as providing some form of dynamic poster or presentation rather than interactive applications. Researchers have proposed a number of alternatives for introducing interactivity, either to turn them into interactive posters or full-scale interactive applications [5]. Although touch interaction is now very familiar to users, there are a number of issues associated with using touch to interact with public displays, ranging from the fact that displays may not be within physical reach to problems of keeping a screen clean for reasons of hygiene and appearance [5].

Since users often have a smartphone on them, or in close proximity, whether at home, at work or on the move [7], there are a number of proposals for using them to interact with large

[1] WebGravity: https://goo.gl/mcO7FC

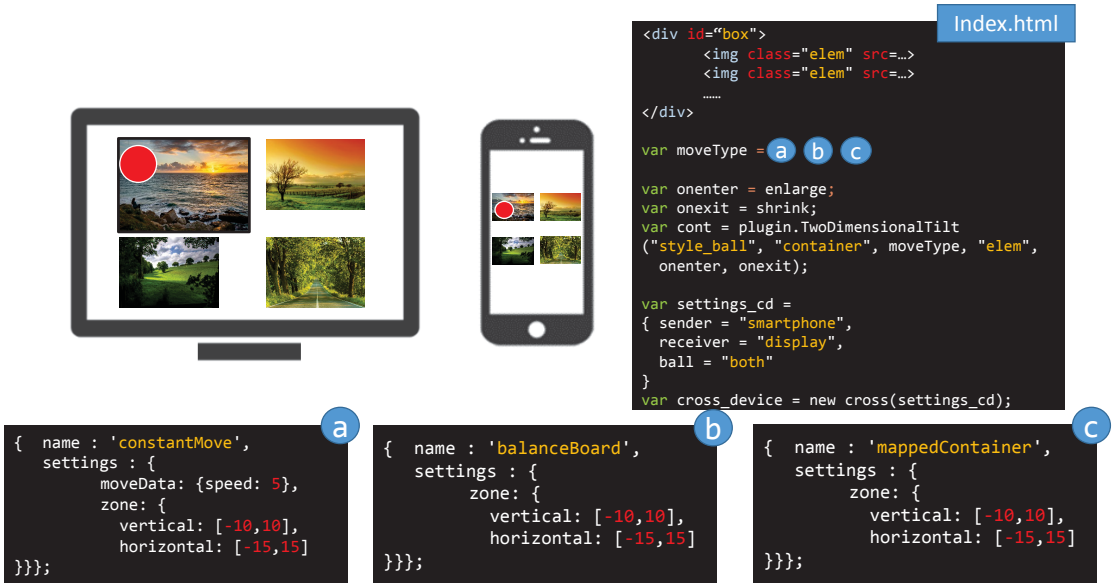



Figure 1: Example use of our framework to continuously control an indicator on a larger device via motion gestures. Developers can decide on the type of interaction by changing the `moveType` variable, in this case: (a) *Constant Move,* (b) *Balance Board* and (c) *Mapped Container*.

displays. Some researchers considered only touch [9], while others investigated mid-air gestures [3, 12] or gaze interactions [17]. In addition, some exploited the fact that smartphones are equipped with motion sensors to explore the use of motion gestures [2, 15, 16].

Using motion gestures such as tilting to control a public display has a number of advantages. First, it does not require any additional hardware either installed on the user or on the display. Second, in contrast to many touch-based interfaces, it does not require the user to look at the smartphone, so they do not need to shift their gaze from the large display [8].

Di Geronimo et. al proposed Tilt-and-Tap, a jQuery framework for the rapid development of tilting interactions [6]. Tilt-and-Tap introduced two main forms of motion gestures: fast movements of the device in a specific direction (jerk tilting) [1], and sustained tilting in a direction to continuously perform some action such as scrolling (continuous tilting) [4]. As previously proposed by Ken Hinckely [10], the framework also offers a combinations of tilting and touch interactions in order to activate and de-activate motion gestures. Tilt-and-Tap was later extended into Cross-Tilt-and-Tap (CTAT) [8] to support interaction across multiple devices. While Tilt-and-Tap offers high level APIs to build motion gestures, it only supports one type of continuous tilting gesture, although researchers have proposed several alternatives for such interactions [2, 14, 18].

For example, Teather and MacKenzie [18] studied velocity-control and position-control techniques. As mentioned in their work, both interactions make use of the orientation of the device but they result in different user experiences. With velocity-control, the more the device is tilted toward a certain orientation, the faster the cursor will move. On the other hand, as also studied by Oakley and O'Modhrain [14], position-control implements a direct mapping between the tilt angle of the device and position of the cursor on the screen. In Teather's study, position-control performed better than velocity-control and, overall, was preferred by users.

Moreover, for each of these techniques, several parameters could influence their behaviours, such as the speed of the cursor, or possible triggers to activate and deactivate motion gestures. Given this multitude of combinations, the implementation of continuous interactions is non-trivial and could provide a barrier to developers extending their applications with tilting gestures.

Motivated by previous work, we decided to design a framework that would not only make it easier for developers to select a particular variant of continuous titing for a given scenario, but to also experiment with these gestures. We then chose to use this framework to investigate alternative continuous tilting interaction techniques for controlling the cursor on a large display with the aim of reducing error rates, while ensuring a good user experience, in order that we could present developers with design guidelines

## WEBGRAVITY

WebGravity is a JavaScript framework for the rapid development of various types of continuous tilting interactions on single device applications. When combined with CTAT [8], it also supports cross-device scenarios where users can interact with other screens by moving their handheld device. We will present the main features of Web Gravity, while showing the three main bi-dimensional continuous tilting interactions supported. To better explain the capabilities of our framework, we will make use of a simple cross-device scenario where users can navigate through a gallery of pictures shown on a smartphone and a large display by simply moving their mobile

device. An indicator – in this case a ball – allows the user to select and enlarge pictures (see Fig. 1).

### Constant Move

As suggested by Boring et al. [2], the simplest way of remotely controlling a cursor on a larger device is to map press downs of physical arrow-keys to a constant motion. With this mapping, the user can move the cursor in only eight directions until the key is released. We applied such a concept in the context of motion gestures where the direction of the cursor is decided by the orientation of the device, as was also done by Boring et al. [2] in their *Tilt* interaction. In contrast to their approach, we kept the movement of the cursor constant as we felt this would allow users to better control the cursor, and hence result in fewer errors, even if it may lead to increased task times[2].

In Fig. 1, we show an example of how developers could use our framework to implement the cross-device application presented above. In the example, the ball is used to select and enlarge pictures with class `elem` inside a container `box`. Using our framework, developers can simply define the movement type and its parameters in the variable `movetype`. As shown in Fig. 1 (a), in the case of the *Constant Move*, developers can also customise the `speed` of the ball by specifying how many pixels the ball moves every 20 milliseconds.

In all of our movements, the key-up behaviour is translated to a *dead-zone* concept. When the three-dimensional orientation of the device is below a certain threshold, no movement occurs and, consequently, the cursor stops. The *dead-zone* refers to a zone around the initial position (when the page is loaded) of the user's device and it can be defined by the developer in the `zone` variable (see Fig. 1) expressed as a range of angles for the x and y axis. The callback functions `onenter` and `onexit` indicate to the developer when the ball enters (or exits) an element. In the example, they are used to enlarge the picture once it is selected and shrink it when deselected.

Moreover, developers can also decide on the look & feel of the ball by passing a customised CSS class, `style_ball` in this example. Once the movement type and various settings are defined, a developer can specify cross-device behaviour by indicating on which devices the cursor will be shown. Here, both devices are used. This is possible thanks to CTAT [8] that collects the ball coordinates on the smartphone and sends them to the right client, in our case the bigger screen. A NodeJS server is then necessary to allow such communication.

### Balance Board

The *Balance Board* movement is inspired by the Marble Maze game[3] where a marble is positioned inside a box with obstacles and holes. The box can be tilted with a knob and the goal is to guide the ball through the maze without falling into a hole. In our case, the device can be seen as defining the area in which the ball may move while the orientation of the device plays the role of the knob. The more the device is tilted in a particular direction, the faster the ball moves.

[2]Constant Move demo page: https://goo.gl/nPVypg

[3]Balance Board demo page: https://goo.gl/rr4dtv

In contrast to *Constant Move*, *Balance Board* allows the ball to move in any direction depending on the tilt angle of the device. Our *Balance Board* movement is similar to the *velocity*-based interaction previously studied by Teather et al. [18]. In their approach, the user's device needs to be positioned parallel to the ground while interacting. Our framework has no such constraint and the user is free to hold the device as they prefer. In Fig. 1 (b), we show how a developer could change the movement type to this form by simply modifying the `moveType` variable, while the remaining code would be the same. With this technique, the speed variable can be omitted since, by default, the movement will directly use the tilt angle of the device.

### Mapped Container

Another method proposed by Oakley et al. [14] for continuously moving an object through the use of motion sensors is to rely on the direct mapping of the orientation of the device to a particular position of the ball in the container. Given the nature of this interaction technique, we called this movement *Mapped Container*[4].

Similar to the *Balance Board* movement, *Mapped Container* interaction allows the user to move the ball in any direction. However, the resulting user experience differs between the two techniques. With *Balance Board*, the tilt angle of the device increases the speed of the cursor, while each rotation angle is mapped to a different position on the page with *Mapped Container*. Similar to *Balance Board*, the speed variable can be omitted and by default the movements of the cursor will depend on the rotation of the handheld device.

### Combining Tilt Techniques

All supported continuous techniques can be combined with various forms of touch interactions. They also have equivalent one-dimensional versions where the movement of the device can be used to scroll a list of elements [14].

## EVALUATION

Similar to the study carried out by Boring et al. [2], we asked participants to use a smartphone to select several target elements shown on a larger display. Participants were allowed to perform the tasks while seated and positioned two metres from the display.

The set-up of the study involved the use of a Huawei smartphone (model: GRA-L09) containing built-in motion sensors, and a 30" HP LCD Screen (model: LP3065) as a public display with a 16:9 aspect ratio. The screen has a 2560 x 1600 resolution, giving a density of 3.96 pixels per millimetre. Participants were seated two metres away from the big screen.

Since mobile devices are usually held in portrait mode, we decided to evaluate the movements in this scenario for both the smartphone and the larger display. Therefore, the area where the ball could move was chosen to be a *portrait* container (width: 1100px, height: 870px), thus smaller than the total dimension of the screen. Each task started with the ball positioned inside an orange rectangle at the centre of the screens on

[4]Mapped Container demo page: https://goo.gl/2Ng6hF

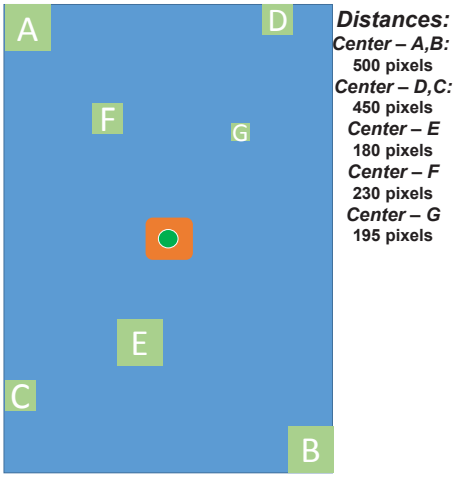



Figure 2: Target elements and their distances to the centre. Large: A, B, E, medium: C, D, F and small: G

the smartphone and the display. The user could start selecting elements after performing a hold tap on the smartphone. At this point, a blue rectangle appeared on the large display and users were required to move the smartphone to reach the target. Once the indicator collided with the rectangle, its colour changed to inform the user that the target was selected. After this selection was successfully performed, users were asked to guide the indicator back to the centre and start a new trial. The time needed to return back to the orange centre box was not included in the total time to complete the task.

Each target had to be selected ten times before a different target was shown on the screen, for a total of seven different target elements (see Fig. 2). We used different target dimensions (50, 75 and 100 pixels), positions (border and non-border) and two selection delays: 0 and 1 second. In the latter, the colour of the target element would first change to orange and then to green once the ball had been inside the element for at least one second. Thus users had to correct overshoots. In the case of no selection delays, the colour of the box immediately changed from red to green, even if the user overshot the target afterwards.

The ball stopped moving once a touch-up event was triggered or if the orientation of the device was inside the *dead zone*. The hold tap interaction allows the user to have better control of the movement and the action of stopping the ball.

Altogether, the study considered the following independent variables:

3 *Interaction techniques (Constant Move, Balance Board and Mapped Container)* 2 *Selection delays (0 and 1 second)* 7 *Targets, selected* 10 *times each*

= 420 trials per participant

To avoid any learning bias, the order of the interaction techniques, as well as that of the targets and the selection delays, were shuffled among users. During the tasks, we logged the ball's trace as well as timestamps for the main events. We also captured the study on video. Moreover, we asked the participants to fill in a questionnaire where they stated their preference for each interaction technique and indicated personal background information. Movements were implemented with our framework

### Results

We conducted the study with 14 participants (5 males and 9 females) with an average age of 30.2 (from 21 to 53 years old). On average, the study lasted one hour. All participants stated that they use their smartphone several times a day. 4 out of 14 participants also stated that they have used mobile tilting interactions at least once in their life, mainly in games.

We compared the three movement types under four measures: the average *time* to select a target, the *throughput*, the number of pixels that the users visited per second, named as *path length*, and the number of *overshoots* that users performed. For each measure, we ran a repeated measurements Anova test and the results are summarised in Fig. 3.

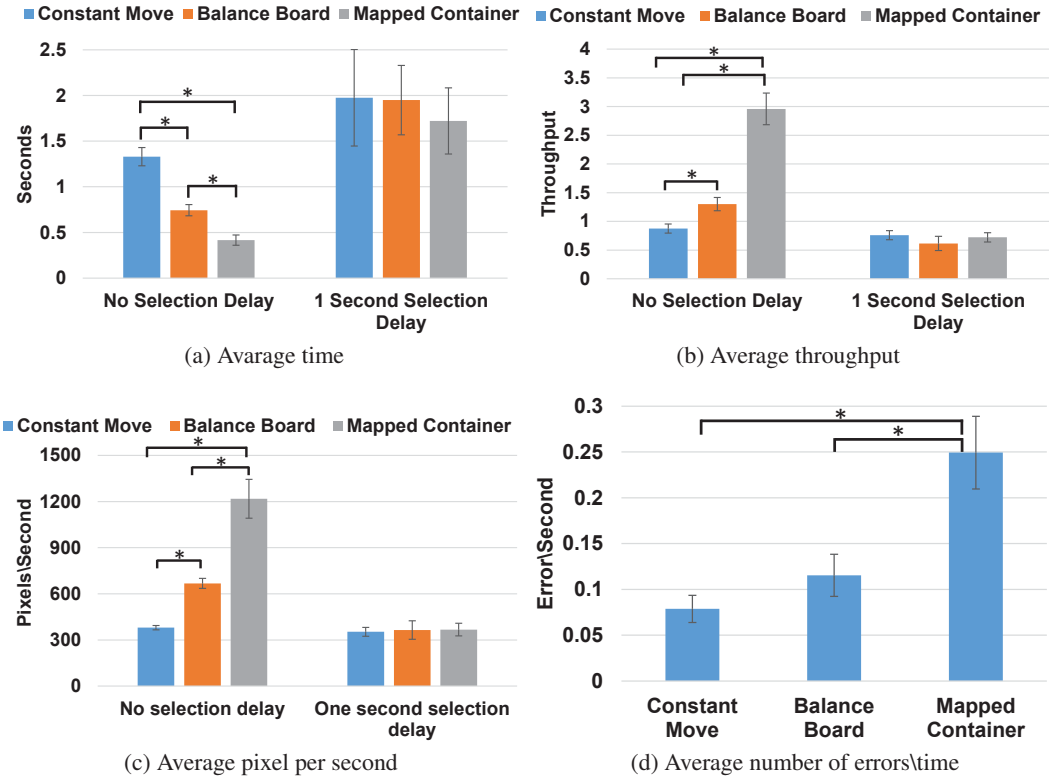
As similarly done by Teather and MacKenzie [18], we calculated the throughput ($TP$) (1) by dividing the index of difficulty ($ID$) by the average movement time ($MT$). As suggested by MacKenzie [11], the index of difficulty is calculated in bits, making use of the distance to the target ($A$) and its width ($W$). For elements A, B, C and D (see Fig. 2), we considered their width as infinite and, therefore, their $ID$ as 0. The width for the other 3 elements was set as their diagonal length.

$$TP = \frac{ID}{MT}$$
$$ID = \log_2\left(\frac{A}{W} + 1\right) \quad (1)$$

Overall, *Mapped Container* performed better among all the movements for no selection delay tasks in terms of time, throughput and path length, followed by *Balance Board* and then *Constant Move*. However, no statistical differences were found among the three movement types when users had to stay on the target for one second.

Generally, participants needed to be more accurate and slower in the delay tasks. Considering all movement types together, participants were on average 45% faster when they did not need to select the target for one second, had higher throughput and visited a larger number of pixels per second. Among all movements, *Constant Move* was the one that suffered the least with the selection delay, while the *Mapped Container*, as well as *Balance Board* movements performed worse when compared with their no delay counterpart.

Regarding the error rates, we calculated the number of times users needed to re-enter the target element due to overshoots [18]. To better show overall performance of the movements, we divided this number of errors by the total task time (see Fig. 3(d)). Since users did not need to correct their selection in no delay tasks, this measurement has been considered only for the selection delay tasks. *Constant Move* and *Balance Board* had similar results regarding this measurement and, overall, a smaller error rate than *Mapped Container* ($F(2,26)=10.112$, $p<0.005$).




(a) Avarage time

(b) Average throughput

(c) Average pixel per second

(d) Average number of errors\time

Figure 3: (a) Time, (b) Throughput, (c) Path Length (px/s) and (d) Overshoots results. Error bars represent standard errors. The * symbol indicates statistical significance among pairs with $p<0.05$.

### Questionnaire

We asked participants to evaluate the three different techniques on a Likert scale from 1 (completely disagree) to 5 (completely agree) in terms of *enjoyability*, *ease of use*, *ease of learning* and *efficiency* in both conditions. We then ran a Friedman test and, when possible, Wilcoxon tests, the results of which are summarised in Fig. 4. Overall, *Mapped Container* and *Balance Board* scored better results than *Constant Move*. While the scores were similar in the delay condition, and *Constant Move* had smaller error rates, users generally preferred *Mapped Container* and *Balance Board* to *Constant Move*. However, *Balance Board* performed differently in the two delay conditions. In terms of enjoyability, *Balance Board* was rated worse than *Mapped Container* but better than *Constant Move*. In no selection delay tasks, no clear distinction was found between those two. Similarly, in terms of efficiency, *Mapped Container* performed better than *Balance Board* in the no selection delay condition, while this distinction was not found in one second delay tasks.

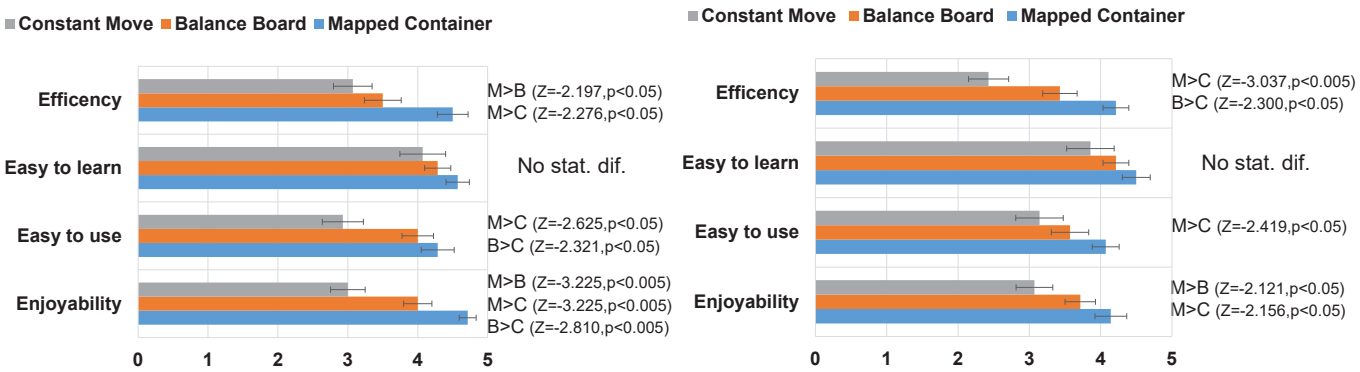
## DISCUSSION

In our study, *Mapped Container* scored the best results in no selection delay tasks. When asked to give general notes about the movement, participants stated the following: *"Fun, makes want to master it"*, *"[...] Very efficient to use."*, *"[...] My favourite option because of its efficiency and predictability"*.

Such findings in a cross-device scenario reflect what was also found by other studies on single device tasks [18, 14]. However, in contrast to Teathers and MacKenzie [18], the performances of *Mapped Container* and *Balance Board* drastically dropped for one second delay conditions and became similar to *Constant Move*. For the two delay conditions, *Constant Move* showed similar results but produced fewer errors than *Mapped Container*. Despite this, participants did not enjoy this interaction and rated *Mapped Container* better in terms of enjoyability, efficiency and ease of use. Based on these results, tilting gestures could be more suited to simulating hovering and scrolling than selection tasks.

Some users liked the combination of tilting and tap hold interactions, stating that this gave them more control over the movement. We also asked participants for their overall opinion about the experiment and in which application they would like to use motion gestures to interact with a large screen. Overall, participants found the study interesting and possible scenarios proposed included presentation applications and performing web browsing or scrolling tasks on televisions.

Based on our experiences of using the framework, we believe that it could significantly aid the research community at large in studies of continuous motion interaction techniques in both single and cross-device applications.




(a) No selection delay

(b) One second selection delay

Figure 4: Average responses to post-task questionnaire for (a) no and (b) one second selection delay conditions. Error bars represent the standard error. Figures also report the Wilcoxon pairwise results between movement types (C) *Constant Move*, (B) *Balance Board* and (M) *Mapped Container*.

## CONCLUSION AND FUTURE WORK

We have presented WebGravity which is a JavaScript framework that can support a wide variety of continuous motion interactions, in particular those proposed by other researchers [1, 16]. In addition, we used the framework to carry out a detailed user study designed to evaluate three different continuous motion gestures. Overall, participants enjoyed the interaction techniques and found them fun and interesting. In the future, we plan to explore multi-user scenarios and study whether motion gestures could improve cooperation among peers when interacting with a public or semi-public screens.

## REFERENCES


1. M. Baglioni, E. Lecolinet, and Y. Guiard. 2011. JerkTilts: using accelerometers for eight-choice selection on mobile devices. In *ICMI*. ACM, 121–128.
2. S. Boring, M. Jurmu, and A. Butz. 2009. Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays. In *OZCHI*. ACM, 161–168.
3. A. Bragdon, R. DeLine, K. Hinckley, and M.R. Morris. 2011. Code space: touch+ air gesture hybrid interactions for supporting developer meetings. In *ITS*. ACM, 212–221.
4. S. Cho, R. Murray-Smith, and Y. Kim. 2007. Multi-context photo browsing on mobile devices based on tilt dynamics. In *MobileHCI*. ACM, 190–197.
5. Nigel Davies, Sarah Clinch, and Florian Alt. 2014. *Pervasive displays: understanding the future of digital signage*. Vol. 8. Morgan & Claypool Publishers. 1–128 pages.
6. L. Di Geronimo, E. Aras, and M.C. Norrie. 2015. Tilt-and-Tap: framework to support motion-based web interaction techniques. In *ICWE*. Springer, 565–582.
7. L. Di Geronimo, M. Husmann, and M. C. Norrie. 2016a. Surveying Personal Device Ecosystems with Cross-Device Applications in Mind. In *PerDis*. ACM, 96–113.
8. L. Di Geronimo, M. Husmann, A. Patel, C. Tuerk, and M.C. Norrie. 2016b. Ctat: Tilt-and-tap across devices. In *ICWE*. Springer, 96–113.
9. R. Hardy and E. Rukzio. 2008. Touch & interact: touch-based interaction of mobile phones with displays. In *MobileHCI*. ACM, 245–254.
10. K. Hinckley and H. Song. 2011. Sensor synaesthesia: touch in motion, and motion in touch. In *CHI*. ACM, 801–810.
11. I. Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.
12. V. Mäkelä, H. Korhonen, J. Ojala A. Järvi, K. Väänänen, R. Raisamo, and M. Turunen. 2016. Investigating mid-air gestures and handhelds in motion tracked environments. In *PerDis*. ACM, 45–51.
13. C. McAdam and S. Brewster. 2011. Using mobile phones to interact with tabletop computers. In *ITS*. ACM, 232–241.
14. I. Oakley and S. O'Modhrain. 2005. Tilt to scroll: evaluating a motion based vibrotactile mobile interface. In *World Haptics*. IEEE, 40–49.
15. K. Pietroszek, J. R. Wallace, and E. Lank. 2015. Tiltcasting: 3D Interaction on Large Displays using a Mobile Device. In *UIST*. ACM, 57–62.
16. M. Rahman, S. Gustafson, P. Irani, and S. Subramanian. 2009. Tilt techniques: investigating the dexterity of wrist-based input. In *CHI*. ACM, 1943–1952.
17. S. Stellmach and R. Dachselt. 2012. Look & touch: gaze-supported target acquisition. In *CHI*. ACM, 2981–2990.
18. R. J. Teather and S. I. MacKenzie. 2014. Position vs. velocity control for tilt-based interaction. In *GI*. Canadian Information Processing Society, 51–58.