CTAT: Tilt-and-Tap Across Devices

Linda Di Geronimo(✉), Maria Husmann, Abhimanyu Patel,
Can Tuerk, and Moira C. Norrie

Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
{lindad,husmann,norrie}@inf.ethz.ch, {apatel,can.tuerk}@student.ethz.ch

Abstract. Motion gestures have been proposed as an interaction paradigm for pairing, and sharing data between, mobile devices. They have also been used for interaction with large screens such as semi-public displays where a mobile phone can be used as a form of remote control in an eyes-free manner. Yet, so far, little attention has been paid to their potential use in cross-device web applications. We therefore decided to develop a framework that would support investigations into the use of a combination of touch and tilt interactions in cross-device scenarios. We first report on a study that motivated the development of the framework and informed its design. We then present the resulting Cross-Tilt-and-Tap (CTAT) framework for the rapid development of applications that make use of various motion gestures for communication between two or more devices. We conclude by describing an applications developed using CTAT.

Keywords: Web interaction framework · Cross-device · Motion sensors

1 Introduction

In 1991, Mark Weiser envisioned a world in which people would be surrounded by devices of different sizes, technologies and goals [1]. Years later, this scenario has become our everyday life. Typically, people now own several personal devices, possibly sharing some of them with family and friends [2]. At the same time, public and semi-public screens are now to be found throughout our places of work and study as well as in public places such as train stations, airports, shopping malls and even in the street. Hence, as Weiser imagined two decades ago and has been confirmed by GSMA Intelligence[1], we are currently living a multi-device era, where the number of mobile devices has surpassed the world population.

In such an environment, Weiser assumed that all of these devices would be interconnected in a vast network to drastically improve the user experience when shifting from one device to another, or when using two or more devices together, despite their intrinsic differences in terms of hardware and goals. While the spread of mobile devices was correctly predicted by Weiser, the vision of a stable cross-device and cross-platform network is still some way off. It remains an

[1] http://gsmaintelligence.com.

A. Bozzon et al. (Eds.): ICWE 2016, LNCS 9671, pp. 96–113, 2016.
DOI: 10.1007/978-3-319-38791-8_6

everyday challenge for users to interact with their set of devices and cross-device applications are still in their infancy [3].

Researchers have explored several ways of supporting interaction in multi-device settings. One is to make use of the motion sensors in mobile devices to pair devices as well as to interact with cross-device applications [4,5]. Specific cases involve using tilting gestures on mobile phones to interact with large screens. This allows users to retain their focus of attention on the large screen while using the mobile phone in an eyes-free manner as a form of remote control.

Despite their advantages, motion gestures have mainly been used in native apps and little attention has been paid to their potential use in web applications as a whole, and cross-device web applications in particular. One reason may be the lack of support offered to web developers as well as portability issues caused by device-dependent sensor APIs. We therefore set out the goal to develop a framework to support the rapid development of cross-device web applications that use a combination of tilt and touch interactions. The resulting Cross-Tilt-and-Tap (CTAT)[2] framework is built on top of two frameworks previously developed in our group: XD-MVC[3] and Tilt-and-Tap [6]. XD-MVC is a framework for cross-device web applications that provides a simple and intuitive API for communication between devices, while Tilt-and-Tap supports the development of web applications that use motion-based interaction. By combining the functionality of both frameworks, there is no need for CTAT developers to handle motion interactions on each device, since they can simply specify one or more senders of the tilting gesture and the corresponding receiver/s.

Before developing the framework, we performed a preliminary study to investigate the potential benefits of using motion gestures in cross-device web applications and inform the design of the framework. For this study, we were able to use the existing Tilt-and-Tap framework with some extensions to cater for a simple cross-device setting. We report on this study in Sect. 3.

The new CTAT framework is able to support a wide-variety of cross-device applications that may span over many different devices. We present the main features of CTAT in Sect. 4 and give details of the implementation in Sect. 5. In Sect. 6, we describe an application that was designed to demonstrate and test the different forms of interaction supported. Finally, we give some concluding remarks and outline future work in Sect. 7.

2 Background

It is now common for mobile devices such as smartphones and tablets to have motion sensors such as accelerometers and gyroscopes that can be used to detect motion gestures. As discussed by Baglioni et al. [7], tilting gestures are a good alternative when touch interactions are not suitable due to users wearing gloves or having dirty fingers. They proposed JerkTilts, a set of toggle gestures where

[2] http://tiltandtap.globis.ethz.ch/ctat.mp4
[3] https://github.com/mhusm/XD-MVC.

users move the mobile device rapidly in some direction. Hinckley et al. [8] further point out that tilting interactions, possibly combined with touch, have the advantage of being eyes-free, single-handed gestures.

Given the potential benefits of motion gestures, we previously decided to study the use of tilting interactions in web applications. This led to the development of Tilt-and-Tap, a jQuery framework for the rapid development of motion-based interaction on the web. Tilt-and-Tap supports combinations of tilt gestures with touch gestures such as tap, double tap and hold tap, together with various feedback modes [6]. Two types of tilting gestures are distinguished: jerk tilting as proposed by Baglioni et al. [7] and continuous tilting where the user interacts with their handheld device by continuously moving it, for example to perform scrolling.

Motion sensors have been applied to cross-device applications for various purposes. Boring et al. [5] employed motion gestures to improve and increase interactions with public screens. In their work, users can remotely control a cursor shown on a large screen by moving their handheld device where the speed of the cursor depends on the tilt angle of the mobile device. The use of mobile phones to interact with large screens was also proposed by Seifert et al. in their PointerPhone project [9]. Making use of a laser pointer mounted on a smartphone, they were able to explore Point-and-Interact techniques where users can point to objects on a large display and then interact with them using their mobile device. For example, a user could rotate a selected object by simply moving their phone. Dachselt et al. [10] have studied the use of jerk and continuous tilting gestures for direct interaction with large displays in detail using applications such as browsing a music library and a map on Google Earth.

Other common uses of motion sensors concern the pairing of devices and sharing of information among devices [4,11–13].

Pering et al. [12] used jerk tilting to play a particular song on a stereo or to turn on lights in a room. While Pering et al. were one of the few to consider jerk tilting in a one-many environment where one handheld device is used to interact with many devices, they do not cover scenarios where multiple smartphones, tablets or screens are involved. As discussed by Kray et al. [14], gestures such as touch or motion interactions may vary depending on several factors including the type of the device which can play an important role. Also, as noticed by Marquardt et al. [15], some interactions may be better suited to smartphone-tabletop communications, while others are better for smartphone-smartphone or smartphone-public display and so on. The amount of possible combinations and scenarios in cross-device applications introduces a challenge that developers need to tackle and could be one of the reasons why cross-device applications are not yet in common usage. In recent years, researchers have tried to solve these problems by proposing a number of frameworks and tools [16–19].

Nebeling et al. proposed XDStudio [16], a visual tool to easily distribute UI elements among devices with the focus on providing different authoring modes for the design of web applications. Chi et al. developed Weave [17] which is a set of high level JavaScript APIs designed to handle different interaction modes

over multiple and diverse devices. Weave offers API support for touch gestures, rotation-change and shake events. For example, multiple users can pair their devices by shaking them at the same time. Without the support of Weave, developers would have to manage the shaking interaction as well as their timestamps on each device. While these works gave us inspiration for our framework, none of them support jerk or continuous tilting interactions.

Moreover, previous research on the use of motion gestures tends to either focus only on native apps, as discussed in our previous work [6], or is very limited in terms of the cross-device settings supported. For example, many researchers have focused on one-to-one scenarios where smartphones are used as remote controls, ignoring the potential use of other mobile devices such as tablets. For these reasons, we decided to take our work on Tilt-and-Tap [6] further by investigating the use of tilting interactions in cross-device web applications. Working with web technologies allowed us to study motion interaction techniques in scenarios where multiple and diverse devices can be involved, giving developers and researchers the opportunity to study and personalize motion interactions in their web application.

3 Preliminary Study on Tilt-and-Tap Across Devices

To better understand the benefits and issues of Tilt-and-Tap style interaction in cross-device settings, we conducted a preliminary study based on a simple, handcrafted cross-device web gallery application that uses tilting gestures on either a smartphone or tablet as a means of interaction. The study involved 12 participants (9 males and 3 females). All participants stated that they to use mobile devices as well as desktop or laptop machines several times a day.

As shown in Fig. 1, users can navigate through a grid of pictures shown on both the smaller screen of a mobile device and a larger desktop screen by continuously moving the mobile device. A ball which plays the role of a cursor is shown on both screens and its movements are influenced by the orientation and speed of the device. When the ball is positioned over an image, that image becomes selected and is highlighted with a red border as feedback on the mobile device and enlarged on the desktop screen. By tapping anywhere in the page, the selected image will be displayed full screen on the desktop screen and its metadata shown on the mobile screen. Users can return to the grid page at any time by rapidly rotating the mobile device counterclockwise. The recognition of tilting gestures is handled by the Tilt-and-Tap framework, while the communication among devices is implemented using Node.js[4] and Socket.IO[5].

To compare tilting interactions to touch gestures, we also developed a touch-only version of the same application. We asked participants to find a particular image in the grid and display it full screen. The task was repeated in four different ways: *smartphone tilt*, *smartphone touch*, *tablet tilt* and *tablet touch*. In the *tilt* versions for both tablet and smartphone, the user could interact with the

[4] https://nodejs.org/.
[5] http://socket.io/.

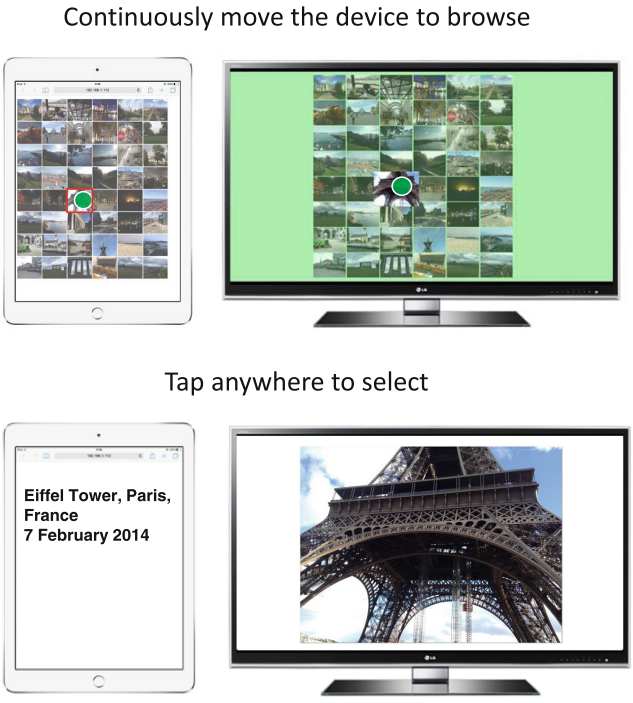

Fig. 1. Tilt-Gallery application and its interaction flow

applications only via motion gestures, while during *touch* tasks, motion gesture weres disabled and only touch interactions allowed.

The grid of images is scaled to display full screen size. We then classify the images according to their position on the mobile screen which reflects how hard it would be to reach them with a thumb. As seen in Fig. 2, and as similarly done by [20], we identified three main areas on the smartphone and four on the tablet. The areas labeled with A have been categorized as hard to reach when the user holds the device in portrait mode, with one hand in the case of the smartphone, and with two hands in the case of the tablet. The B zones were identified as easier to reach. However, participants were unaware of this categorization. The setup of the study is shown in Fig. 3. The devices involved in the study were: a 24 inch. TV as the desktop display, an iPad Air, and an iPhone 6. In addition, a laptop was used to show the image that the user was required to select.

Users were given a brief explanation of the study and a training phase on a test page, before starting the tasks. They first had to select nine images using the *smartphone touch* version with the images distributed evenly across the three areas labelled A or B. The same task was then repeated on the *smartphone tilt* version. Similarly, for the tablet, the users had to select eight different pictures, with two in each of the areas labeled A or B, performing this first for the touch-only version and then for the tilt-only version. To avoid a learning effect on

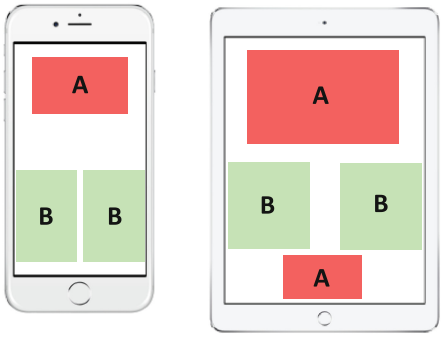

Fig. 2. Identified areas on smartphone and tablet

Fig. 3. Study environment

the position of the pictures, a completely different set of images was shown for each task. Moreover, half of the participants started with the smartphone tasks, while the rest started with the tablet tasks. At the end of the study, participants filled out a questionnaire that involved background information as well as qualitative questions about the study. We recorded the study on video in order to analyse particular user behaviours during the tasks and keep track of the time to complete a task.

As can be seen in Fig. 4, participants enjoyed using tilting interactions. More than 80 % of users agreed or strongly agreed that motion gestures were enjoyable on the smartphone, while around 70 % of participants found it enjoyable on the tablet.

Generally, motion sensors performed better on smaller mobile devices with 70 % of participants finding it easy or very easy to use them on smartphones, with the corresponding figure for tablets being 50 %. This trend is also mirrored in the question where participants were asked to rate efficiency of motion gestures. 40 % of participants did not find tilting interactions particularly efficient on tablets, while only 20 % were of the same opinion for smartphones.

The velocity of the indicator was the same for both the mobile and tablet tasks. Therefore, to move the ball to a particular position, the user either had to use a higher tilt angle on the tablet or tilt the device for a longer period of time. Some participants perceived the *tablet tilt* task to be slower than the *smartphone tilt*. This factor could have been one of the reasons why users preferred tilting interactions on smartphones. However, tilting interactions on tablets have their advantages since some participants felt that motion gestures were more suited to the iPad where they appreciated the ease of controlling the ball on the larger screen.

When compared to touch only gestures, around 50 % of participants found tilting gestures on both tablet and smartphone (50 % and 59 % respectively) comparable to, or less demanding than, touch. The average time to complete a task was similar for all four versions at around 7 seconds per image. Moreover,

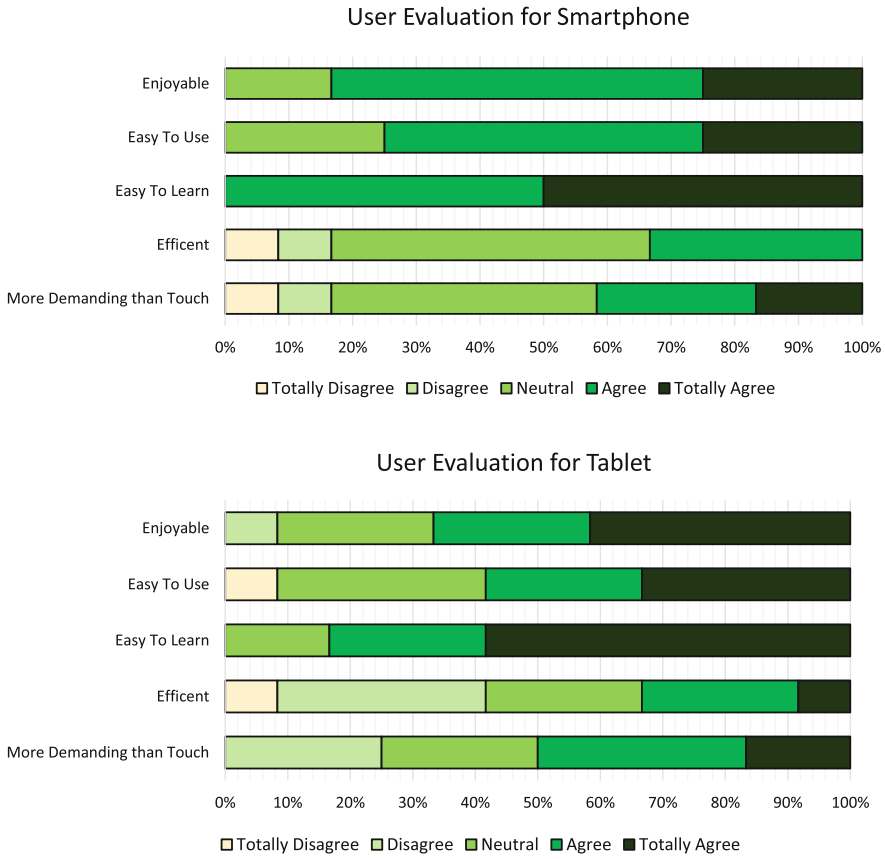

Fig. 4. User evaluation for tilting interactions on smartphone (top) and on tablet (bottom)

there was no significant statistical difference between times to select images located in different areas. We note however that the main focus of the study was not to directly compare motion-based interaction against touch, but rather to receive feedback in order to improve and design a version of Tilt-and-Tap better suited to cross-device applications.

One interesting finding concerns the focus of attention of the participants during tasks. In the case of tilting interactions, most users concentrated exclusively on the desktop screen which was not true in the case of touch. One participant commented on this behaviour by saying: *"[...] during the tapping I could not use the TV at all since I would have to search for the image twice."* During the *tablet touch* task, only one participant started by paying attention to the TV screen but, after few moments, he changed his focus to the mobile device and said: *"Why look at the TV, I need to select it from the iPad anyways."*.

These findings motivated us to pursue our research on the use of motion sensors in cross-device applications. Based on the feedback from users regarding the limitations as well as the potential of our tilting interactions, we were able to design and develop our cross-device framework CTAT which pays more attention to the intrinsic differences among devices and adjusts tilting interactions accordingly.

4 The CTAT Framework

Taking into account user feedback as well as our experience of using Tilt-and-Tap in a cross-device environment, we developed the new CTAT framework, which is specifically designed for the rapid prototyping of applications with cross-device tilting interactions.

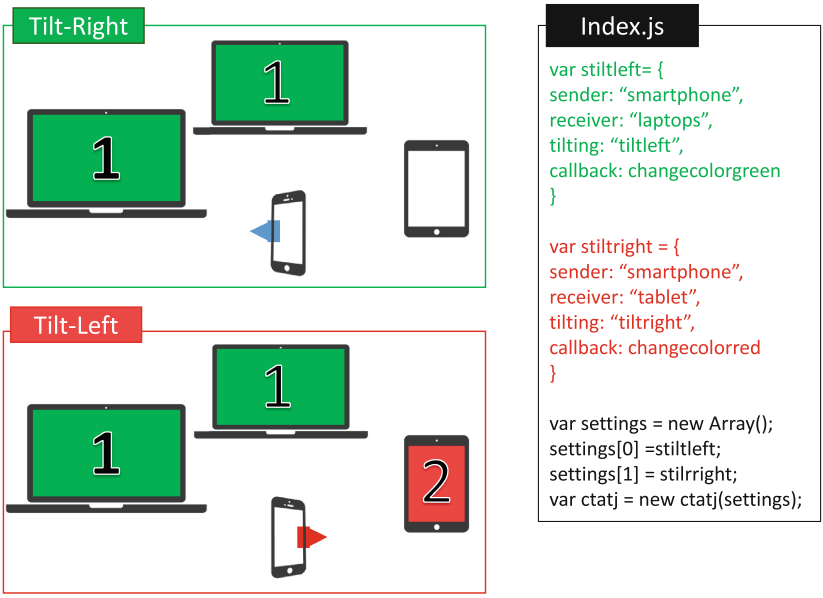

Fig. 5. Cross-device jerk tilting interactions and their implementation

CTAT offers support for the two main forms of motion-based interaction used in Tilt-and-Tap, namely jerk and continuous tilting. Since many applications tend to use only one of the two forms of tilt interaction, we decided to actually produce two variants CTATJ (CTAT-Jerk) and CTATC (CTAT-Continuous). The resulting reduction in size and complexity of the framework required in many cases, can significantly improve performance. However, in cases where both forms of tilting interactions are used in an application, the two frameworks can be used together.

In the example shown in Fig. 5, jerk tilting gestures are used to change the background colour of other devices connected to the same web page. By rapidly rotating their smartphone to the left, the colour on the two laptops will change, while rapidly tilting to the right will change the colour on the tablet. The code to achieve this behaviour using the CTATJ variant of the framework is shown on the right of Fig. 5.

As the names suggest, the variables `stiltleft` and `stiltright` correspond to the tilt left right and tilt left gestures. These two objects contain all the information that CTAT needs to manage motion interactions among devices. The `sender: "smartphone"` setting indicates the actor in the tilting interaction, while `receiver: "laptops"` and `receiver: "tablet"` specify the target devices for the tilt left and tilt right gestures, respectively. The function specified in the callback option will be executed on the receivers whenever the corresponding tilting interaction is performed on the sender.

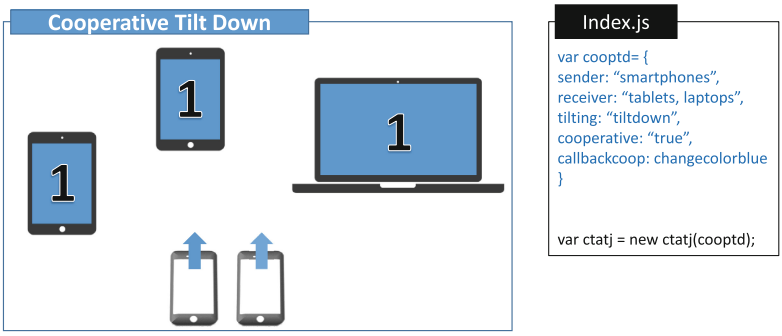

Fig. 6. Cooperative jerk tilting gestures and its corresponding implementation

CTATJ also offers cooperative tilting gestures. As shown in Fig. 6, users can simultaneously perform a particular motion gesture to interact with other devices. In our example, if two or more smartphones simultaneously perform a tilt down gesture, this will modify the background colour of all connected tablets and laptops. To distinguish between cooperative and non-cooperative jerk tilting gestures, the `cooperative` option inside the settings object has to be set to `true`. Moreover, the `callbackcoop` option defines the callback function for only cooperative executions of the gesture.

A cross-device continuous tilting example implemented with CTATC is shown in Fig. 7. Similar to the web gallery application developed for our preliminary user study, users can interact with a large screen by moving their smartphone. The red ball shown on the laptop simulates a cursor. When the ball is over one of the elements, that element becomes selected and is enlarged on the laptop screen. As with CTATJ, developers can define a sender, one or more receivers, possible touch interactions and a callback function for when an element becomes selected. In addition, with the `ball: "laptop"` option, the developer specifies

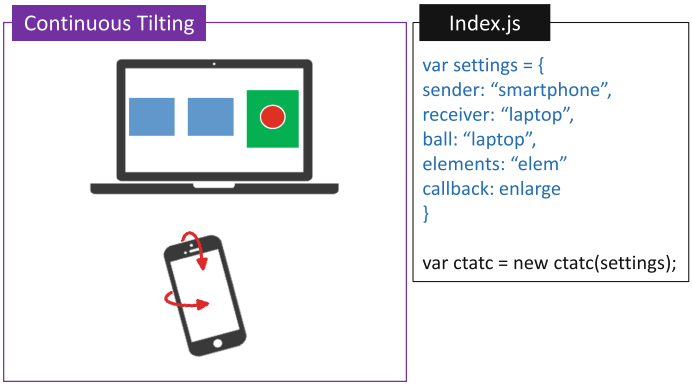

Fig. 7. Cross-device continuous tilting interactions and their implementation

on which device the ball will be shown, in this case, the laptop screen. Any number and types of devices can be specified under this option.

While it is very easy to specify interactions using these settings variable, the lines of code necessary to define motion gestures increases with the number of interactions that the developer wants to recognize on each device. For this reason, we have developed a visual tool that automatically generates the CTAT objects.

The tool is shown in Fig. 8. Developers can add devices as well as creating new communications among them in step 1. When a new connection has been added, the corresponding settings can be modified in the *Connection Manager* menu shown in step 2. If developers save the connection (step 3), icons will be displayed for each device involved in the communication (step 4). The direction of the arrow indicates if the device is a sender or receiver of that specific tilting interaction. The tool is implemented as a web applications in HTML, CSS and JavaScript.

5 Implementation

From our preliminary study, we recognised two main problems of using Tilt-and-Tap for cross-device applications. The first of these concerned the development process and the second the lack of support for adapting interactions to different devices, especially in continuous tilting interactions.

Managing tilting interactions on every device required us to develop tedious and long code, making our applications more prone to errors. Socket.IO allowed communication among devices, but, for each interaction, it also required an exchange of messages between the sender, the server and finally the receiver. Moreover, continuous tilting interactions in cross-device applications revealed some challenges that were not originally considered by Tilt-and-Tap.

Cross-Tilt-and-Tap was developed specifically with cross-device applications in mind and therefore a primary goal was to overcome these issues. As previously

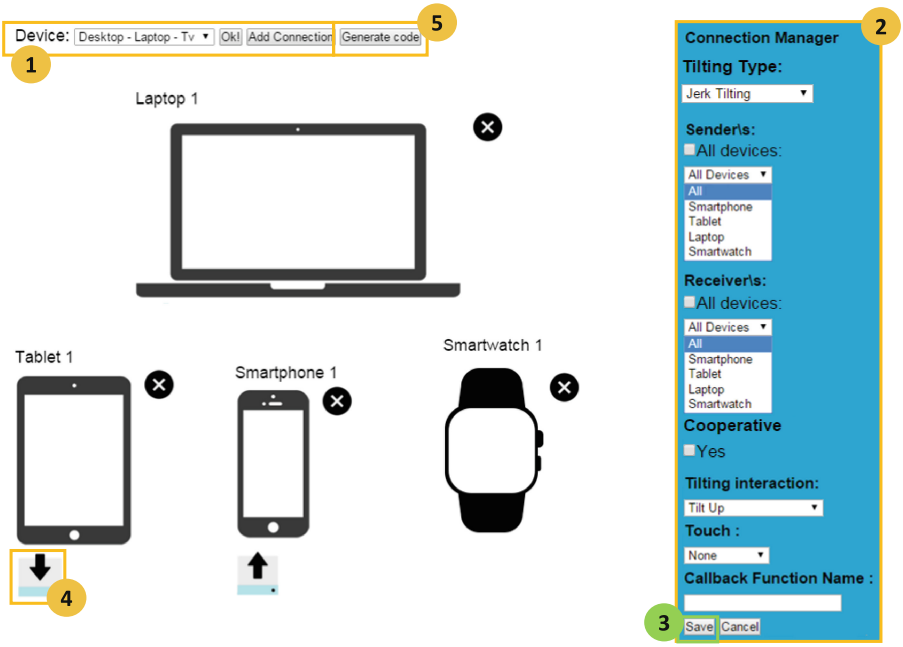

Fig. 8. Visual tool for the generation of CTAT objects

mentioned, one of the first decisions made was to split responsibilities for jerk and continuous tilting between the two sub-frameworks CTATJ and CTATC.

To share real-time messages among devices, CTAT uses the communication module of XD-MVC, a cross-device framework previously developed in our group. XD-MVC includes a Node.js server, and uses Socket.IO and Peer.js for cross-device communications. When supported by the browser, peer-to-peer communication is used, resulting in lower latency which is crucial, especially for continuous tilting. As a fallback, client-server communication can also be used and this allows us to support a wide range of devices. XD-MVC abstracts this mechanism from the developer, thus allowing direct communications among clients to be easily managed.

We note that since CTAT can involve many different devices, developers have to include the client as well as the server side of our framework in their web application. A Node.js installation is then required on the machine that will act as the server. Other than this, no particular installations are required on the client nor on the server.

Figure 9 indicates some of the APIs supported by CTAT and its two sub-frameworks CTATJ and CTATC.

We currently support four categories of devices: laptop-screens, tablets, smartphones and smartwatches. These devices are recognised by making use of user agent information. Developers can use these types of devices to specify a particular

CTATJ and CTATC

Option	Description	Default Value	Possible Values
sender	Indicates the sender(s) of the interaction. Multiple senders available only for CTATJ.	null	"all"; "tablet(s)"; "smartphone(s)"; "smartwatch(es)" "laptop(s)" and combination
receiver	Indicates the receiver(s) of the interaction.	null	"all"; "tablet(s)"; "smartphone(s)"; "smartwatch(es)" "laptop(s)" and combination
touch	Indicates if the jerk tilting interaction needs to be performed with a touch gesture	null	"tap"; "double tap"; "tap hold"
callback	Callback function executed on receiver when jerk tilting interaction performed or when ball selects an element	null	any existing function

CTATJ

Option	Description	Default Value	Possible Values
tilting	Indicates the jerk tiling interaction	null	"tiltup"; "tildown"; "tiltleft"; "tiltrigh"; "tiltse"; "tiltsw"; "tiltnw"; "tiltne"; "tiltclockwise"; "tiltcounterclockwise"
cooperative	Indicates if the jerk tilting interaction is cooperative	false	true, false
callback coop	Callback function executed on receiver when cooperative jerk tilting interaction performed	null	any existing function

CTATC

Option	Description	Default Value	Possible Values
ball	Indicates if ball is visible and where.	null	""; "sender"; "sender-receiver"; "receiver"
elements	Indicates the class name of elements that the ball will select	null	any existing class

Fig. 9. List of main APIs supported by Cross-Tilt-and-Tap

client or set of clients for their tilting interactions. As seen in Fig. 9, CTATJ allows many-to-many communications, meaning that the same tilting interaction can be performed by more than one client and these events will trigger the execution of a callback function on one or more receivers. Developers can indicate a set of devices by using its plural form such as "laptops", "smartphones" and so on. Similarly, to target one specific client, developers can simply use the singular form of the device name. In this case, our framework will assign the first client connected to the page of the set specified by the developer. When the selected device disconnects, the next client will be assigned to the interaction.

In contrast to non-cooperative gestures, when a developer indicates that a particular communication is cooperative, all the senders specified should perform the tilting interaction simultaneously.

While CTATJ allows many-to-many connections, CTATC only considers one-to-many communications since it is counterintuitive to have more than one device able to remotely control a cursor on another client.

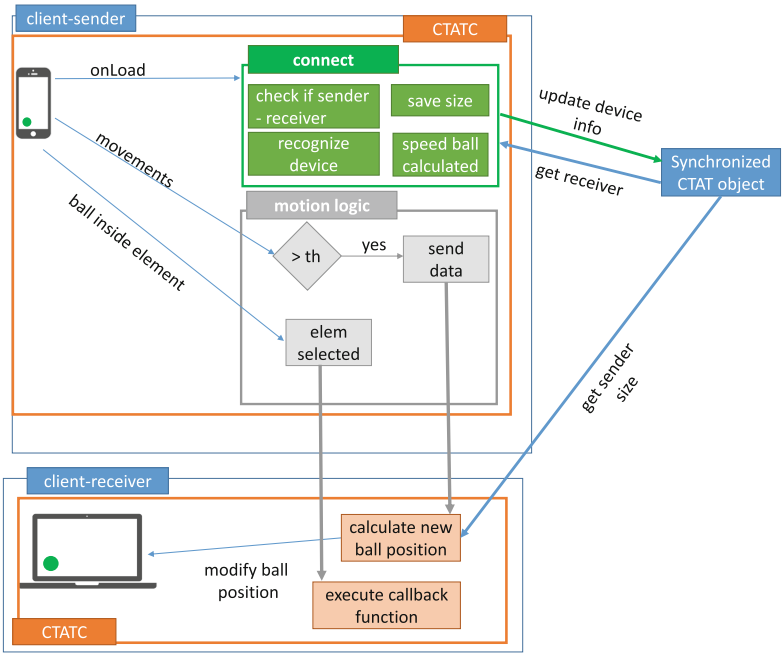

Fig. 10. Example of CTATC execution flow

To explain in detail how continuous tilting interactions among devices are managed we show the execution flow in Fig. 10. In our example, a smartphone device is used to remotely control a large screen represented as a laptop. We assume that the larger device is already connected to the web page, and the mobile device initiates a connection. Every time a new client connects, the device type is first checked. If it is involved in a tilting interaction as a sender or receiver, we save its width and height. All this information is stored in an object that is shared among all the connected devices with our framework ensuring that these shared objects are synchronised across devices. For example, if a device that is currently a sender for a tilting interaction disconnects, CTAT needs to modify the corresponding shared object and assign a new sender to the tilting interaction if possible. As indicated in Fig. 10, whenever a device is assigned to be the sender for a tilting interaction, the speed of the ball cursor is calculated dynamically depending on the screen size of the sender device. By default, the speed of the ball increases slightly as the dimension of the sender screen size increases. This parameter can also be adjusted by the developer using the corresponding setting.

In Tilt-and-Tap, the movement of the ball is managed on the sender side. In the case of CTAT, the framework has the role of communicating changes to all the receivers. For performance reasons, we first filter the movements performed by the mobile device on the client side and only send the new ball coordinates to the receivers if the movement performed is larger than a threshold. At this

point, the receiver has all the necessary information to update the position of the ball. The new position is calculated by multiplying the ball position sent from the sender with the quotient of the receiver and sender width and height. This proportion allows the user to have a good match between the two devices despite their differences in resolution.

When the ball moves inside one of the elements, the framework wraps the event triggered on the sender side and sends this information to all the receivers involved in the interaction. We note that while the motion logic module is only present in CTATC, the overall behaviour of the jerk tilting version of our framework, CTATJ, is similar.

6 Demonstrator Application

To define, demonstrate and test the capabilities of the CTAT framework, we designed an advanced gallery application that featured the various forms of interaction and device configuration that we aimed to support.

The application was designed to engage users with semi-public and public displays throughout our university and promote social interaction among viewers. As mentioned earlier, many cross-device applications have been based around the use of mobile phones as a sort of remote control device for large displays. We wanted to take this idea further by considering multiple displays and multiple mobile devices and allowing them to take different roles. Also, there should be some form of interaction between mobile devices as well between mobiles and the displays.

The resulting application, called aCrossETH, was inspired by 500px[6] a well known image sharing website for photographers. Similar to 500px, users upload images and other users can then like or favourite them, with the most highly-rated images making it onto a "Popular" page[7]. All tilting interactions included in the aCrossETH application were developed making use of CTAT, while the GUI was implemented with HTML, CSS and JavaScript.

Figure 11 shows the three display categories in aCrossETH: slideshow-screen, voting display and mobile devices. The slideshow screen shows the six most popular images uploaded by users where the popularity takes into account the number of likes and favourites together with its freshness. In the example setting shown in Fig. 11, this role is taken by a projected display in a social area. All the most recent uploaded images are shown in a grid layout on one or more voting displays.

By scanning a QR code shown on both the slideshow and voting displays, users can interact with voting displays using their mobile device. The first connected mobile user is assigned the role of controller which allows them to browse images on the voting display by simply swiping left and right on their mobile device. Additional information about a selected image will be shown on all of the connected mobile devices which have the role of viewers. The selected image will also be enlarged on the voting display. A different version of the same application

[6] https://500px.com/.

[7] 500px recently merged likes and favourites into a single Twitter-style heart.

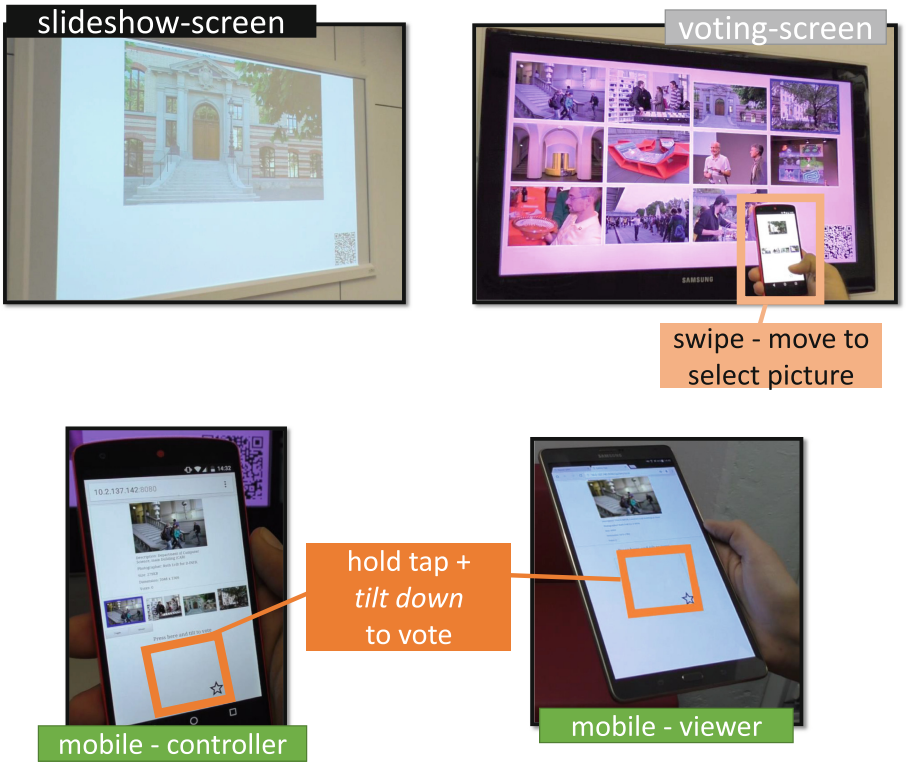

Fig. 11. Overview of aCrossETH on the three different categories of screens: slideshow, voting and mobile displays

allows the controller to browse the gallery of images using continuous tilting. In this version, the ball is shown only on the voting screen.

Detailed views of the mobile user interface for the controller and viewers is shown in Fig. 12. Users can vote for the current selected image using a hold tap on the rectangular area at the bottom of their mobile screen and rapidly tilting the device down. While the viewer device can only see the current selected image, the controller has an overview of all images visible on the voting screen. To improve user engagement, if the vote gesture is performed simultaneously by two or more devices, the number of votes added is doubled. This means that, the more users cooperate, the greater the chances that the images they vote for will be among the most popular shown on the slideshow screen.

By tapping on the upload button displayed on mobile devices, users can upload new images. Once they have selected an image from their local gallery, the system shows a preview of the selected image in full screen mode. To receive immediate feedback, users can decide to share the image with other connected mobile devices by simply performing a tilt left gesture as shown in Fig. 13. At this point, all the connected devices can see the new image and choose to like or

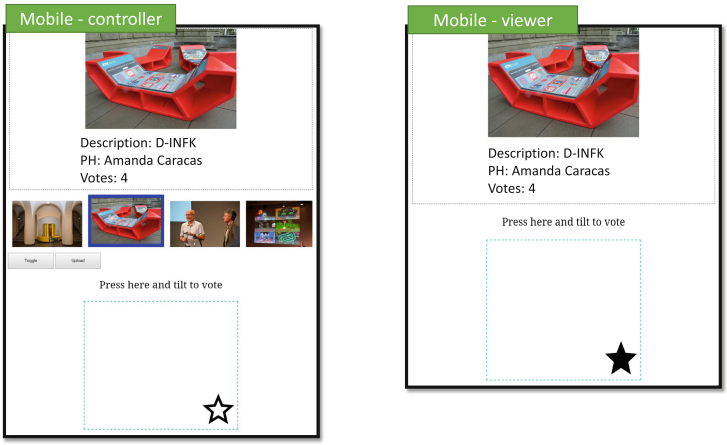

Fig. 12. User interface of aCrossETH on mobile controller and viewer devices

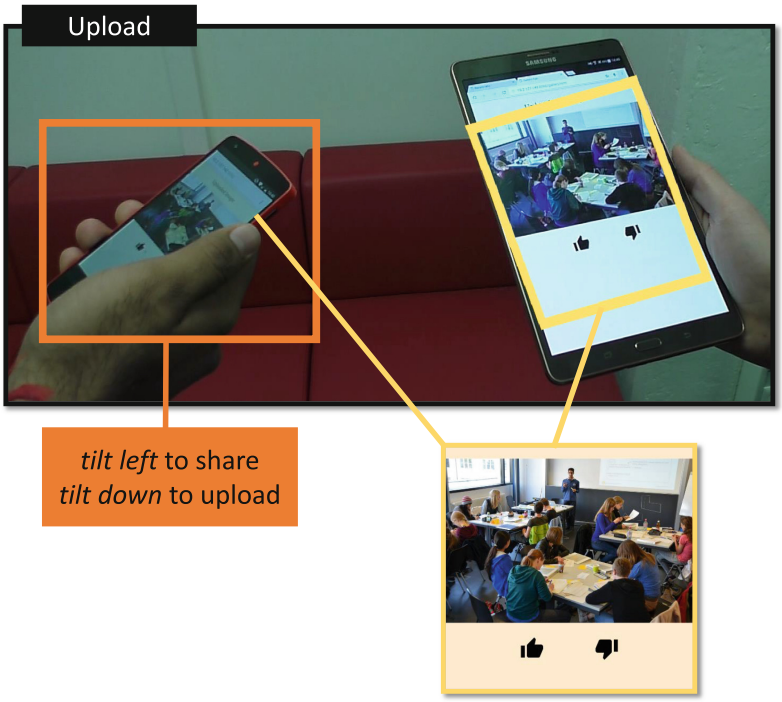

Fig. 13. User interface of aCrossDevice when user uploads a new image and shares it with other devices

unlike it. The owner can see how users have voted on their mobile device and, based on that, decide whether or not to go ahead and upload the image to the

system. Once their image has been uploaded, they can view the voting screen by performing a tilt down gesture.

Developed in parallel to the framework, this application gave us useful insight into the kinds of interactions that the framework should support as well as testing both their use and implementation. In turn, the framework made it easy to experiment with alternative versions of the application and different kinds of interactions.

Finally, we note that the use of motion-based gestures enables users to interact with the application in an eyes-free manner while mobile within the university environment. Further, as shown in the results of our preliminary study, motion gestures are often perceived as more enjoyable than the sole use of touch interactions when a mobile device is paired with a larger display. Taken together, we believe that such an application could be integrated into a pervasive display system to encourage user engagement.

7 Conclusion

We have presented CTAT, a framework that supports the rapid prototyping of cross-device web applications that employ motion-based interaction. We believe that such a framework is necessary to push forward research in motion-based interaction on the web in general, and in cross-device applications in particular, by supporting experimentation with novel applications and modes of interaction.

Now that we have the framework, we plan to experiment further with multi-user, multi-device settings, investigating the potential benefits of motion-based interaction for interacting with public displays, sharing information and also generally moving information between devices. We also plan to take the work on visual development tools further, by building on ideas from previous work in end user development [21].

References

1. Weiser, M.: The computer for the 21st century. Sci. Am. **265**(3), 94–104 (1991)
2. Facebook: Finding simplicity in a multi-device world, March 2014. https://www.facebook.com/business/news/Finding-simplicity-in-a-multi-device-world
3. Santosa, S., Wigdor, D.: A field study of multi-device workflows in distributed workspaces. In: Proceedings of the UbiComp. (2013)
4. Yatani, K., Tamura, K., Hiroki, K., Sugimoto, M., Hashizume, H.: Toss-it: intuitive information transfer techniques for mobile devices. In: CHI 2005 Extended Abstracts on Human Factors in Computing Systems, pp. 1881–1884. ACM (2005)
5. Boring, S., Jurmu, M., Butz, A.: Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays. In: Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7, pp. 161–168. ACM (2009)
6. Di Geronimo, L., Aras, E., Norrie, M.C.: Tilt-and-Tap: framework to support motion-based web interaction techniques. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 565–582. Springer, Heidelberg (2015)

7. Baglioni, M., Lecolinet, E., Guiard, Y.: Jerktilts: using accelerometers for eight-choice selection on mobile devices. In: Proceedings of the 13th International Conference on Multimodal Interfaces, pp. 121–128. ACM (2011)
8. Hinckley, K., Song, H.: Sensor synaesthesia: touch in motion, and motion in touch. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 801–810. ACM (2011)
9. Seifert, J., Bayer, A., Rukzio, E.: PointerPhone: using mobile phones for direct pointing interactions with remote displays. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013, Part III. LNCS, vol. 8119, pp. 18–35. Springer, Heidelberg (2013)
10. Dachselt, R., Buchholz, R.: Natural throw and tilt interaction between mobile phones and distant displays. In: CHI 2009 Extended Abstracts on Human Factors in Computing Systems, pp. 3253–3258. ACM (2009)
11. Aumi, M.T.I., Gupta, S., Goel, M., Larson, E., Patel, S.: Doplink: using the doppler effect for multi-device interaction. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 583–586. ACM (2013)
12. Pering, T., Anokwa, Y., Want, R.: Gesture connect: facilitating tangible interaction with a flick of the wrist. In: Proceedings of the 1st International Conference on Tangible and Embedded Interaction, pp. 259–262. ACM (2007)
13. Hassan, N., Rahman, M.M., Irani, P., Graham, P.: Chucking: a one-handed document sharing technique. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5727, pp. 264–278. Springer, Heidelberg (2009)
14. Kray, C., Nesbitt, D., Dawson, J., Rohs, M.: User-defined gestures for connecting mobile phones, public displays, and tabletops. In: Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 239–248. ACM (2010)
15. Marquardt, N., Hinckley, K., Greenberg, S.: Cross-device interaction via micro-mobility and f-formations. In: Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, pp. 13–22. ACM (2012)
16. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M.: Interactive development of cross-device user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2793–2802. ACM (2014)
17. Chi, P.Y.P., Li, Y.: Weave: Scripting cross-device wearable interaction. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 3923–3932. ACM (2015)
18. Krug, M., Wiedemann, F., Gaedke, M.: SmartComposition: a component-based approach for creating multi-screen mashups. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 236–253. Springer, Heidelberg (2014)
19. Kovachev, D., Renzel, D., Nicolaescu, P., Klamma, R.: DireWolf - distributing and migrating user interfaces for widget-based web applications. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 99–113. Springer, Heidelberg (2013)
20. Wolf, K., Henze, N.: Comparing pointing techniques for grasping hands on tablets. In: Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services, pp. 53–62. ACM (2014)
21. Paternò, F., Santoro, C., Spano, L.D.: Model-based design of multi-device interactive applications based on web services. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5726, pp. 892–905. Springer, Heidelberg (2009)