Tilting Single Handheld Mobile Devices

Solange Karsenty and Avi Moraly

Hadassah Academic College, Computer Science Department
37 Hanevi'im, 91010, Jerusalem, Israel

Abstract. Gestures are becoming common across many devices, but still remain in the category of games and entertainment. In this paper we have explored the use of gestures on mobile devices, in particular the action of *tilting* on an iPhone. Our goal is to enhance the user experience and increase productivity, assuming, as it is often the case, that the user is using the device with a single hand. In this case, the hand holds the device and the thumb is used to scroll and select. Using the thumb for selection is error prone. Our goal is to reduce the use of the thumb for operations such as scrolling and selecting, by tilting the device to perform similar operations. We have implemented and tested a prototype application and in this paper we present our results. Our conclusion is that standardization of gestures is a basic need in software development kits for mobile devices.

1 Introduction

The popularity of mobile devices, such as the iPhone, iPad, iPhone and the Android, have brought new styles of interaction and user experience. Early work by Rekimoto [1] proposed mapping motion input to tilt. Since then, mobile devices have considerably evolved. The iPhone for example is equipped with sensors that provide data relating to orientation, position and movements. These sensors allow the recognition of motion and gestures that have become more and more popular, especially in gaming applications. Researchers have explored the use of gestures for a variety of tasks such as map navigation [2] panning or zooming [1]. Meanwhile, vendors are expanding the functionality of software development kits that allow increased experimentation of new gesture-based applications.

End users are developing the dexterity needed for touch screens and small displays, and it is becoming common to see users manipulating these displays using only one hand. In one-handed operation the thumb is used as a pointing device, and used to perform common operations such as scrolling and selecting, but often this results in many input mistakes. Since the device is being held in the palm of the hand, it becomes unstable, and the thumb is needed to serve two contradictory roles: pointing and holding (stabilization). Due to these two different roles the result is often imprecise pointing, or unstable holding which in the worst case results in dropping the device. The use of fisheye interfaces [4] in single handheld devices is one attempt to overcome this problem.

Karsenty S. and Moraly A..
Tilting Single Handheld Mobile Devices.
DOI: 10.5220/0003705601160121
In *Proceedings of the 2nd International Workshop on Software Knowledge* (SKY-2011), pages 116-121
ISBN: 978-989-8425-82-9

1.1 Tilting

The iPhone, and other devices are equipped with an accelerometer (plus magnetometer and gyroscope) that allows users to tilt the device and interact by controlling orientation and movement of the device. As shown in the figure below, the SDK provides measures of change of velocity along three axes, allowing detecting motion in any direction:

- *X = Roll.* X values vary from 0.5 (rolled all the way to the left) to -0.5 (rolled all the way to the right).
- *Y = Pitch.* from 0.5 (the headphone jack straight down) to -0.5 (the headphone jack straight up).
- *Z = Face up/face down.* face up (-0.5) or face down (0.5).

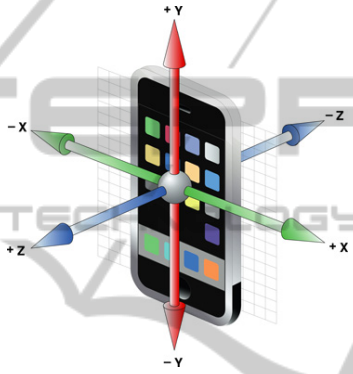

Fig. 1. The iPhone accelerometer axes.

Early work by Rekimoto [2] discussed the power of navigation and scrolling by using motion input. Since then researchers have conducted various experiments, and the recent popularity of smartphones and tablets has triggered the development of numerous entertainment applications based on advanced touch screens and sensors. Tilting was used to allow a user to change screen orientation [5] or to input text [6]. The traditional event model of window toolkits has been enriched with touch events that capture various states: a *tap* is when we touch the screen and immediately lift a single finger. A *gesture* is a sequence of events that happens from the time we touch the screen with one or more fingers, until we lift our fingers off the screen. The events generated by the gestures are handled through three methods (touch, move, release), implementing a basic three state machine.

2 The Application

Our primary goal is to enhance the user experience. Therefore we developed an application that combines two standard functions, an address book and dialer, based on interactive tilting gestures. The application includes browsing and searching through a contact list, dialing and accessing the dialing log.

2.1 The Gestures

Gestures on mobile devices have been characterized [7] and are continuously evolving. Since our primary goal is to improve single-handed input, we are interested in gestures that can be achieved while holding the device in one hand. Tilting is the most natural gesture as opposed to other gestures requiring the use of fingers. The iPhone can detect tilting in any direction. We use it to scroll in two ways:

1. Fine scrolling: when more than three items are shown and the gesture will usually scroll more than one item. For example, when going through a list of contacts, tilting up/forward scrolls up, tilting down/backwards scrolls down.
2. Coarse scrolling, or scroll and select: when less than three items are shown, the current, the next and previous (see Figure 2). In this case the gesture unambiguously selects the current, next or previous item. For example, the application contains several screens virtually ordered that can be accessed by tilting right (next screen) or left (previous screen). On each screen, a preview icon indicates the adjacent screens (see figure 1 the contacts and phone log icons).

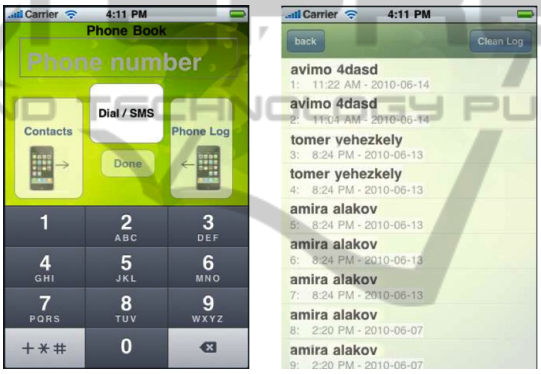

Fig. 2. The home screen: tilting right from the main screen brings to the phone log (coarse scrolling). The phone log list is scrollable by tilting up and down (fine scrolling) while left/right tilting selects and dial the current selection.

Both techniques can be combined. Coarse scrolling usually fits horizontal lists (left/right), while fine scrolling is used vertical (up/down). Coarse scrolling is used as a mean to select an item. For example, the user will move up and down the device to go through a long list of contacts, and then tilt to the right the device to select and access the current contact.

3 User Tests and Results

We carried out a user evaluation to assess the usability of tilting. We have chosen to test our application with both novice and common users of the iPhone. Our test was based on two tasks: searching for a contact and dialing, searching through the call history and dialing. We compared our application against the iPhone standard applications. The results have shown the following:

1. Novice users are very uncomfortable and less efficient with tilting, at least in the beginning. We found that tilting is actually too responsive and hard to control for novice users.
2. Common users, who already used tilting with other applications, quickly mastered the scroll and select operations. Those who have a lot of experience with gaming hence had already become proficient in tilting, were the most efficient.
3. In general, our application performs better over time or after a period of training. However when the scrolling list containing a hundred or more entries, most user prefer a direct search by typing. This is actually true for standard scrolling as well.

In our first experiments we quickly discovered few problems:

1. Unwanted scrolling when holding the device in a non vertical position
2. Unwanted scrolling with small movements while holding the device in neutral position (initial natural position).
3. Difficulty to control the scrolling speed, therefore difficulty to access the target
4. Loss of visibility on the screen when the device orientation is over 45 degrees (with respect to the initial orientation). See figure 3.

Fig. 3. The natural position of the device provides the widest vision angle. When tilting forward to scroll up, objects appear smaller and light reflections reduce readability.

The first problem was easily fixed by calibrating the accelerometer. By using the initial natural position of the device as the neutral position, we automatically calibrate the application. The user can use the device for example while lying down. We solved the second problem by adding hysteresis, which is triggering the scrolling command only above a minimal initial input value. In fact hysteresis is implemented for most input devices although it is hardly noticeable.

The last two problems are related, in order to scroll faster, users increased the device orientation, but then lost visibility due to the angle and reflections on the display. Initially we placed an upper value limit on the scrolling speed, but users would still orient the device more than what was needed. We then realized that feedback was missing in order to let users know when the maximum speed was reached.

We also found that setting default values was a hard task: the more users used the tilting, the faster they needed to scroll. Ideally, these parameters should be available in the standard configuration parameters of the device, similarly to mouse tracking speed.

We found difficult to concentrate on our user experiment, and we tried out different ideas, this was mostly because each step had to be implemented as a very low-level of hardcoded software. For example, the scrolling acceleration had to be handled by the application outside the event object, while we consider it as a first class proper-

ty of the event. It is important to note that a tilting widget may or may not have a graphics representation depending on the required user feedback: the fact that the data is scrolling is already a form of feedback. A traditional scrollbar has buttons for actions. These buttons are not necessary when tilting. If needed, the display of a tilting widget can be limited to a simple moving bar that shows current position within the list.

Another difference is the notion of acceleration; the traditional scrollbar acceleration is internally managed and the user has no direct control on it. Tilting is more complex since the acceleration is directly controlled by the gesture. Nevertheless gestures can be characterized and wrapped as event objects and graphics widgets with properties and methods. These can be fully integrated in current widget toolkits.

4 Gestures and Software Toolkits

Mobile devices are now commonly equipped with multi touch screens and gyro sensors. Therefore users are becoming familiar with the use of gestures. Gestures include for example: tapping, pinching in and out (usually for zooming in and out), panning or dragging, swinging in any direction. The iPhone OS for example, provides a UITouch event class, but it does not provide a built in generic gesture event that would allow for instance the recognition of a "Z" gesture event.

Early work such as GT2K [8] used speech recognition theory to implement a gesture recognition toolkit. Echtler et al. [9] proposed a layered software architecture to improve the cross platform development of multi touch interfaces. The architecture includes a widget layer that integrates with existing toolkits. It does not provide a model for integrating the various motion gestures at the toolkit level. We have the theory and knowledge to recognize gestures, but there are still a number of issues that make it hard to conduct experiments and establish a cross platform extensible software model for gestures.

No Standard. Vendors have different software models that make it impossible to even implement the same gestures across devices. Only some gestures such as pinching are becoming popular and available in most platforms. Note that the W3C DOM Level 3 event model does not support gesture events at all.

Gesture Recognition Engine. We need to experiment with new gestures; therefore we need to create new kinds of events. Such events must use an engine to perform real time recognition. When the recognizer is implemented at the application level, it performs with poor response time. Response time is critical for smooth scrolling and user control. Such engines should available in the software toolkit to insure real time recognition.

Hardware Evolution. Hardware is quickly evolving, mobile devices are equipped with new sensors and the combination of these sensors create new sophisticated interactions. For example, the current ability to recognize a simple shaking movement might soon become a more refined event with directions: left/right movement shaking, up/down movement shaking, or twist (wrist rotation) shaking. It is hard to anticipate software needs and build cross platform applications when basic hardware sensors evolve so fast.

Nevertheless, researchers have been exploring new interaction techniques based on experimental device sensors. For example, early work by Harrison et al. [10] showed how pressure sensors could be used to detect in which hand the user is holding the device.

5 Conclusions

We have built an application that demonstrates the power of tilting on mobile devices. While researchers have proposed many interesting gestural interaction techniques, we believe there are many more gestures that can improve the user experience, using sensors and multi touch displays. While user interface toolkits and window systems have been stable for years, the fast expanding development of mobile applications using gestures requires that we revise the event model. Events are now more complex: these are made of series of events that require interpretation by a dedicated engine. Software toolkits do not support yet these gestures as abstractions and developers spend a considerable amount of time to implement ad-hoc code.

If gestures can be specified as a language and software toolkits provide a gesture recognition engine, gestures can be created as high-level event that can be processed by the application. With the fast evolution of hardware, it would be then easier to experiment with gestures and define an evolving standard. Such standard would help research developers to perform usability tests and comparative evaluation of different gesture based interfaces.

References

1. Rekimoto J.: Tilting Operations for Small Screen Interfaces. Proceedings of UIST (1996) 167-168
2. Hinckley, K., Pierce, J. Horvitz, E.: Sensing Techniques for Mobile Interaction. Proceedings of ACM UIST (2000) 91-100
3. Weberg L., Brange T., Hansson, W.: A piece of butter on the PDA display. CHI '01 extended abstracts ACM (2001) 435–436.
4. Karlson A-K, Bederson B., SanGiovanni J.: AppLens and launchTile: two designs for one-handed thumb use on small devices. Proceeding of CHI'05, ACM (2005)
5. Hinckley K., Pierce J., Sinclair M., Horvitz E.: Sensing Techniques for Mobile interaction. Proceedings of UIST '00, ACM (2000) 91–100
6. Wigdor D. , Balakrishnan R.: TiltText: Using Tilt for Text Input to Mobile Phones. Proceedings of UIST '03 ACM (2003) 81–90
7. Ruiz J., Li Y., Lank E.: User-defined Motion gestures for Mobile Interaction. Proceeding of the 2011 conference on Human Factors in Computing Systems (2011)
8. Westeyn T., Brashear H., Atrash A., Starner T.: Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. Proceedings of the 5th international Conference on Multimodal Interfaces (2003)
9. Echtler F., Klinker G.: A Multitouch Software Architecture. Proceedings of NordiCHI'08 ACM (2008)
10. Harrison B., Fishkin K., Gujar A., Mochon C., Want R.: Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. Proceeding of CHI'98 ACM (1998).