MOBILE TOUCH INTERFACES FOR THE ELDERLY

Roger Stone
Loughborough University
Loughborough, LE11 3TU, England
R.G.Stone@lboro.ac.uk

ABSTRACT

Elderly people are not averse to buying and using electronic gadgets. However regarding certain devices there is a persistent complaint about the "buttons being too small". Therefore the arrival of mobile touch devices like the iPhone and iPod Touch should be able to circumvent that problem because the button size and arrangement is under software control. However these devices have some accessibility issues which are identified. The accessibility issues stem from the one-size-fits-all concept. A solution is proposed which involves having a range of interface styles. A new user gesture called the *shake* is proposed to switch between interface styles. A separate investigation is made into the different possibilities for free-text entry.

KEYWORDS

Touch screen, user interface, mobile device, elderly, accessibility, gesture.

1. INTRODUCTION

With the elderly set to form a larger and larger percentage of our population it is increasingly important that attention is paid to supplying them with usable digital devices. It is clear from the success of the Nintendo DS-Lite with Brain Training in the older market that elderly people are not averse to buying and using portable electronic gadgets. However regarding mobile phones, cameras and remote controls there is a persistent complaint about the "buttons being too small". Therefore the arrival of mobile touch devices like the iPhone and iPod Touch should be able to circumvent that problem because the button size and arrangement is under software control. But are these devices universally accessible, in particular for the elderly? The stretch and pinch gestures using finger and thumb which increase and decrease the magnification of the page view are very welcome and can bring text up to a readable size. However a jump to a new page will bring up a new page 'squashed' to fit onto the screen and so the expansion gesture must be repeated on the new page. The user input of free text has a particular problem. A pop-up keyboard occupying half the screen area is used and the size of the keyboard cannot be adjusted by the stretch gesture. If the elderly user for reasons of eyesight or finger control (or a combination) cannot discriminate between the 26 alphabet keys on this soft keyboard then this device has a serious accessibility flaw.

Researchers have proposed profile-based solutions for accessibility [e.g. Stone, Dhiensa & Machin 2006] where the final rendering is adjusted according to font-size, font-colour preferences, etc. of the user. It has been pointed out that a single user may need different profiles at different times for example to cope with the onset of tiredness. These solutions can in principle be applied to small mobile touch devices but there are still problems. One problem is that these profile-based adjustments work best on weakly styled pages where the author has not tried to force the page to have a particular 'look' (elastic interfaces [Griffiths 2004][Stone 2006]). The more pixel-based styling that the author uses, the worse the visual effect becomes of re-styling the page with a user-profile. Unfortunately the arrival of the iPhone/iTouch has produced an outpouring of so-called iApps heavily optimized for the small screen which therefore causes profile-based adjustments to behave poorly. There are significant gesture possibilities for an elderly person using a mobile device [Shannon 2008]. The iPhone/iTouch is regarded as having gone beyond the traditionally accepted WIMP interface [Jacob, *et al*., 2008].

This paper offers two particular avenues of approach for the accessibility of these small mobile touch devices. One is shown in the context of an extension to an existing user-interface library to provide a range of profiles rather than a single interface. The other is an approach to solve the free text input problem.

2. MOBILE TOUCH INTERFACES

The devices being considered are the iPhone and the iPod Touch which have identical touch screen interfaces. For brevity the term iTouch will be used to refer to either device. The screen size is 3" by 2" and the iTouch operates either in portrait or landscape mode with a motion sensor which allows the device to automatically sense the orientation and offer differential styling. The on-board applications mimic Apple desktop applications but their interfaces have all been redesigned to work with the touch interface and to utilise some novel input gestures. A high quality, standards conforming web browser (Mobile Safari) is provided. Of the standard user interface input elements (button, menu and text input) the menu and text input have been specially implemented. Touching a menu on a web-page causes a half-height, full-width selection menu to pop up which responds to flick-up and flick-down gestures and touch selection. Touching a text input box brings up a half-height, full-width soft keyboard with 26 alphabetic keys and 8 other control buttons.

The iTouch has two orientations (portrait and landscape) which can be detected via Javascript and different stylesheets automatically applied for each orientation. Rather than using alternative stylesheet technology, attribute selection styling is used. Different values for attributes are set dynamically when an orientation change occurs. Variations of styles with different style selectors are provided (e.g. body[orientation='portrait'] {*portrait style definition*} body[orientation='landscape'] {*landscape style definition*}). When an attribute of a node in the document is changed this can mean that a different style now matches and the browser is obliged to render the effect on the screen. Most of the conditional programming to date just relies on testing whether the device is in portrait or landscape orientation. However there are in fact two landscape orientations depending on whether the device is turned from portrait to landscape in a clockwise or anti-clockwise direction.

2.1 Styling for accessibility

Conventionally the font-size is set slightly higher for landscape as there is more screen width and thus people with poor vision might turn the device to landscape for better readability of text. But what if this single alternative is not enough. For accessibility it may be necessary to have a range of styles at different sizes. In order to allow a range of styles (i.e. more than 2) an extension of this idea has been tried. This involves a new gesture, christened the *shake*. The style required is obtained by a *shake* of the phone. Two different *shakes* of the phone are defined, a shake-*up* (a rotation from portrait, clockwise to landscape and back again) and a shake-*down* (similar but starting with an anti-clockwise rotation). The shake gesture can be detected by javascript and a global variable (*shakes*) can be incremented for each shake-up and decremented for each shake-down. Instead of using the code body.setAttribute('orientation', 'portrait'), the code body.setAttribute('orientation', 'portrait'+shakes) is used to set the orientation attribute to portrait0 or portrait1 or portrait2, etc. Now a different style (or style overlay) can be defined for every different value of the global variable (e.g. body[orientation='portrait0'] {…}, body[orientation='portrait1'] {…}, etc.). It is anticipated that styles associated with larger numbers will render larger views on screen by using increased font-size for example. Every time the user *shakes* the iPhone the style changes. When the user performs a shake-up gesture the font-sizes, etc. get larger. When the user performs a shake-down gesture the font-sizes, etc. get smaller.

There is an emerging standard user interface library for the iTouch called iUI [Hewitt 2007]. This provides a CSS file and an accompanying Javascript file. By using the portrait and landscape styles in this library as a guide, the various numbered styles can be defined as described above. Within the Javascript code the function orientationChange() must be altered to maintain two more global variables currentOrientationAngle and the shakes counter using the built-in property window.orientation. Finally the (re-)assignment of the orientation attribute of the body which happens every time the orientation changes must be altered to involve the shakes count.

Note that only the orientation attribute of the body tag has been singled out for change and not other nodes in the document. In practice this has been found sufficient as if other tags (for example a list item) need a different style then these can be triggered from the body change by using a style selector of the form

body[orientation='portrait2'] > ul > li {*list item style*}

By this method all iApps developed using this library can be made more accessible. The switching between styles is by the new gesture called a shake. It is hoped that this will prove to be a natural and easily remembered gesture as it comes close to the idea of shaking a non-functioning device in order to try and 'get it to work'.

3. KEYBOARD ENTRY

Free-text input on keyboards without a QWERTY keyboard can be very frustrating. Trying to rename a recording on a domestic dvd player using an up, down, left, right, OK remote control typically requires driving a cursor step by step around an alphabet display. To some people sending an SMS text message using a conventional mobile phone is very tedious. The obvious method for keyboard entry on a touch screen device as used on many PDAs is to draw a QWERTY keyboard and make each letter into a button. However other designs for touch screens have been tried with some success notably MessageEase [Nesbat 2003]. For finger touch devices it is crucial to know the minimum button size and spacing that can be successfully employed and whether this varies with age [Jin et. al. 2007]. Working at the minimum button size (allowing the maximum number of buttons on display) is not necessarily the best solution if other factors like the speed of operating the interface are considered important [Sears et. al.1993].

Free text data entry on the iTouch is achieved by a pseudo keyboard which takes up approximately half the screen or an area approximately 2" wide by 1.5" high. This offers all 26 alphabet letters along with 8 other buttons. This means that there are 34 targets in 3.0sq.in. thus every target occupies on average 0.088sq.in. or 57sq.mm. Now this has obviously been proved acceptable for a large range of users otherwise the device would not have had the success it has. And there is a safety net. Each letter key expands when touched so that the user can see more easily what they have 'hit'. If this is not the key they wanted then they can slide their finger to another key as it is the key being touched when the finger is lifted off that is selected. There is another facility for accented characters that if a key is held for long enough (1 sec) then all the different accented versions of the letter are offered. However once the user has got this far they cannot then slide to a totally different letter. This (incorrect) letter has to be accepted and then the delete key used to remove it.

But what if the user's acuity of vision and hand-eye coordination do not allow them to reliably target an area 7mm square? There is no alternative. Such a user is effectively barred from using such a device. But is this acceptable? The computer science community is supposed to be offering accessible interfaces. So a solution ought to be offered.

3.1 Twelve way selection

As already noted a common interface for entering text is via the 12 button (0-9*# layout) SMS interface found on all button-based mobile phones. However this is a sequential interface because after one press of (say) the 2 button it is only known that the user wants to enter either an 'A', 'B', 'C' or '2'. The actual choice between the four characters is a secondary decision. If a text prediction scheme is being used the user is encouraged to go on to type the next letters hoping that the prediction scheme will guess the word intended. If a prediction system is not being used, multiple key presses cause a cycle through the available options. However there is another complication involving timing which is an accessibility issue. There is a timing interval defined such that if the inter-press interval is less than the timing interval a repeat keypress is taken to be a different choice of character on this particular key, but if the inter-press interval is bigger than the timing interval then it is taken to mean that one text character has been chosen and a new text character input sequence is commencing. This rather complicated process is a direct result of the fixed, 12-button, hardware keyboard. On a touch screen interface a much simpler process is easily implemented. The familiar 12-button keyboard can be displayed at first but after a touch of (say) the ABC2 button a mini keyboard with just 4 buttons can pop up so that the user can complete the character input with a 1-out-of-4 choice. In this way a 1-

out-of-26 choice has been replaced with two choices in sequence (1-out-of-12, 1-out-of-4) and the buttons can be proportionately bigger allowing greater accessibility if required.

3.2 Binary selection

Shannon's theory of information [Shannon 1948] shows us that the amount of information needed to enter a single character choice out of n possible characters is log2n if all the characters are equiprobable and -sigma (p_i * log 2 p_i) where p_i is the probability of character i if they are not equi-probable. The worst case of a person's discrimination is when they can only make a 1-out-of-2 or binary choice. Information theory provides a lower bound on the number of such choices that need to be made. The entropy of a set of 26 equi-probale characters is $\log_2 26$ or approximately 4.7 so the theory suggests that an alphabet letter entry can be accomplished in 5 binary steps. Even using realistic probabilities for the letters (e.g. those used for Morse Code [Wikipedia]) gives an entropy of 4.22 so on average more than 4 binary choices will be needed. This leads to an interface following ideas similar to the Huffman coding algorithm [Huffman 1952] where the first step is to divide the characters into two sets of approximately equal total probability and the first choice is to say in which of the two sets the desired character lies. The process continues until a set is chosen containing only one character. The computed probability algorithm would have as its first choice "is your letter one of ETAINO or not?" as these 6 letters have a combined probability of approximately 0.5. The equi-probable algorithm would have as its first choice "is your letter one of ABCDEFGHIJKLM or not?". Given the relatively small difference between the number of binary choices needed to choose a letter in the two algorithms, it would seem likely that the best choice would be the equi-probable algorithm as it requires less effort to answer the questions. Moving away from binary choices, information theory also shows that better results can be obtained by making 1-out-of-k choices with k>2 and indeed better results can be anticipated the bigger k becomes.

3.3 A range of text interfaces

So much for the theory. Does this allow for realistic interface building? What if the user performs a shake-up gesture when a 26-way text input interaction is on screen? What lower discrimination interface should be offered? Or looking at things another way - imagine that there is a game that the user can play to find out what level of discrimination they can manage. The screen is progressively divided up into 2 (2x1) buttons, then 4 (2x2), then 9 (3x3), then 16 (4x4), etc. until the user cannot reliably manage the discrimination between the buttons. The idea would then be to decide on a figure *n* such that the system will never require from the user anything more precise than 1-out-of-n discrimination in any interface in any application for that user. However this figure *n* should not be regarded as fixed. Experience shows that in different parts of the day, or under different lighting conditions, etc. the discrimination level may change and hence the shake gesture should also be able to modify the text input interface. The challenge is to make a range of text input interfaces that suit the shake gesture but with the differences between interfaces less dramatic than the three described above.

4. CONCLUSION

Some of the most recent high-performance, mobile, touch-screen devices have proved very popular even to older people. However the one-size-suits-all interface of a device like the iTouch, although very carefully engineered, does pose a few accessibility problems for older people. This is a shame because the very nature of the device allows it to be very flexible. In examining the interface a novel gesture has been developed for switching styles to improve readability. With this gesture, christened a shake, backed up by javascript programming and css styling, the user is able to step through graded interfaces until text becomes readable. This technology can readily be incorporated in an existing interface library. It is hoped that this will prove to be a natural and easily remembered gesture as it comes close to the idea of shaking a non-functioning device in order to try and 'get it to work'.

On the user-input side of the interface the free text input has been singled out as the most problematic. In the search for a text entry system that requires less discrimination than the QWERTY keyboard, a variation of the 4x3 button mobile phone interface has been described which suits touch screen interfaces. This brings the discrimination required down from 1-out-of-26 to 1-out-of-12. The ultimate binary interface is also discussed which only requires a 1-out-of-2 discrimination. The challenge is to make a range of text input interfaces that suit the shake gesture but with the differences between interfaces less dramatic than the three described above.

ACKNOWLEDGEMENT

Funding for this work was supplied by the Research School of Informatics.

REFERENCES

Griffiths, P., 2004, "Elastic Design", http://www.alistapart.com/articles/elastic

Hewitt, J. 2007, "iUI – User Interface (UI) Library for Safari development on iPhone", http://code.google.com/p/iui/wiki/Introduction

Huffman, D.A, 1952, A Method for the Construction of Minimum-Redundancy Codes, *Proceedings of the IRE*, Volume: 40, Issue: 9, pp 1098-1101

Jacob, Girouard, Hirshfield, Horn, Shaer, Solovey, Zigelbaum, 2008, "Reality-Based Interaction: A Framework for Post-WIMP Interfaces", Proceedings CHI 2008, pp201-210

Jin, Plocher, Kiff, 2007, "Touch screen user interfaces for older adults: Button size and spacing", Universal Access in Human Computer Interaction: Coping with Diversity (part 1), *Springer Lecture Notes in Computer Science*, vol 4554, pp 933-941

Nesbat, S.B. 2003, "A system for fast, full-text entry for small electronic devices" (MessageEase), *Proceedings ICMI*, ACM Press, vol 4, No 11

Sears, Revis, Swatski, Crittenden, 1993, Investigating touchscreen typing: the effect of keyboard size on typing speed" *Behaviour & Information Technology*, vol 12, pp17-22

Shannon, C.E., 1948, "A Mathematical Theory of Communication", *Bell System Technical Journal*, 27, pp. 379–423 & 623–656, July & October, 1948.

Shannon, M.M., 2008, "This menu has changed", Communications of the ACM, May 2008, Vol. 51, No. 5, pp19-21

Stone, R.G., 2006, A lightweight Web GUI specification and realization system and its impact on accessibility, *Proceedings of the 2nd International Workshop on Automated Specification and Verification of Web Sytems (WWV 2006)*, Paphos, Cyprus, pp37-42

Stone, Dhiensa & Machin, 2006, Profile-based Web Document Delivery, *Proceedings of the 2006 ACM Symposium on Document Engineering*, Amsterdam, pp 215-217

Wikipedia, "Letter frequencies, Morse Code", http://en.wikipedia.org/wiki/Letter_frequencies