Lecture Summary 4: Mobile I/O Interfaces

The primary means of interaction on the mobile is the touch-screen, which is used both for output (visual) and input (touch-based). The touch-based control is an interesting evolution because it has replaced both the conventional keyboard and mouse that we have been used to with personal computers. Screens are based on the same screen technology as desktops but are smaller. Resolutions have improved on mobiles, but we will continue to be restricted by their smaller physical size. The main challenge of mobile I/O is to adapt our applications by taking account of the limitations, and some advantages, offered by the new interfaces.

	HTC Tattoo	*Nexus One*	*iPhone 4*	*Samsung Galaxy Tab 10.1*
Resolution	240x320	480x800	640x960	800x1200
Pixels	0.08MP	0.38MP	0.61MP	1MP
Aspect	1:1.33	1:1.67	1:1.5	1:1.6
Density	180ppi	254ppi	326ppi	149ppi
Size	53x71mm	56x94mm	59x89mm	113x180mm

Figure 1: Screen properties vary widely between devices

Screens

Details in figure 1 show that screens may vary considerably with respect to several properties. Sizes and resolutions range from the tiny HTC Tattoo, to being significantly dominated by larger existing tablets such as the Galaxy tablet. These two devices in particular both operate on Android, which immediately demonstrates some concern of potential differences in viewing experience.

It is worth noting that the HTC Tattoo represents the extreme, and most smartphones screens now support a screen-size in a range above this. In general both screen-density and sizes have been seen to improve steadily (see figure 2), there are for instance about 5 times as many pixels on the iPhone 4S compared to the original from 2007. The noticeable deviation of typical density in the iPhone 4S (326ppi) is claimed to be a 'retina display'. Which is taken to mean that a human should not be able to distinguish between individual pixels at an operational distance (threshold is about 290ppi).

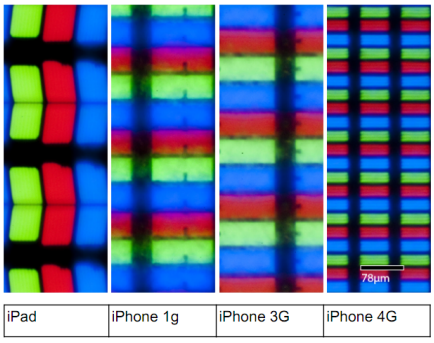

Figure 2: Pixel densities have improved for each succession of the iPhone. The iPad tablet has a lower resolution because operating distance is larger.

Another difference to considerater are variations in screen technology. On most displays pixels are tiled as squares and composed of three OLEDs (red, green and blue), such as in figure 2. An alternative technology is the PenTile RGBG display, a Samsung patent, used on most Samsung phones as well as the Nexus One. On these screens, instead of square tiles, the pixels are arranged in alternative rows of small green OLEDs, and larger red/blue ones. The green are smaller because humans can distinguish shades of green better than red or blue. Blue OLEDs in general require a higher operating current which has an affect on their lifetime, they come dimmer compared to the others pixels over time. To balance this effect they are made slightly larger so that they emit the same light at lower current and thereby decay at the same rate. To this effect all OLEDs are shipped at slightly higher intensity than considered optimal, because they eventually drop to a level that users find more comfortable. The PenTile requires a conversion for displaying bitmaps, which is good for photographs and natural scenes, but reduces the contrast of pixelated graphics such as text. This is a slight consideration for developers, we may for example choose to increase the sizes of some fonts to obtain better performance across devices.

Considering the properties in figure 1 further, we should note that devices are manufactured in virtually every possible configuration within these ranges. The implication on different screens-characteristics on the developer is that results may vary widely between devices. Anything positioned absolutely may be OK on one mobile, but may be out of bounds on a smaller screen. Using absolute dimensions on any element will mean they will vary in physical size in proportion to the screen density. Different screen-ratios means it's not possible to solve screen-layouts just in terms of relative scaling, which in principle is a

good solution for applications such as graphics based games. Just relying on scaling may on large screens make information appear sparse in some applications.

Android provides several tools to help the developer. One such aid is generalised densities and screen-sizes that correspond to predefined ranges. Small screen-sizes, for instance, are between 2-3 inches, and low densities are between 100-150 dpi. In response one might for instance use a two column layout on a large display, and a single column on a smaller one. There are still variations within each category, so fluid layouts (particularly useful for interfaces) are usually employed, that are analogous to the the way html structures layout on websites. These are based on a hierarchical model of containers and elements, most often specified with relative widths that can expand in relation to each other. An element in support of this approach are images defined as *9patch*, which can be set to expand relative to a bounding box. If constant sizes is required on the other hand, which may be appropriate of images such as icons, Android has the option of density independent pixels which consistently scale to a physical size of $1/160$ of an inch. A sophisticated approach towards display-variation are Android *Fragments*, that can be used where applications cleanly distinguish between different activities such as navigation and display of content. The desired effect is to automatically structure the layout of application to show multiple items on larger display, compared to being divided in different views on a relatively smaller one.

Touch-based Input

Touch-based input is radically different to what we've been used to with the keyboard and mouse, and has important implications in our applications design.

The technical implementation of touch detection are resistive and capacitance based. The resistive type depends on the surface being compressed which changes the impedance of embedded conductors, resistance is measured along the sides of the phone to determine the x,y coordinate of the input. The resistance based screens are cheap and power efficient, but have poor clarity and prevents the use of screen protection. The capacitive technology works by changes in the magnetic fields that the capacitors produce. The sensors can detect changes in fields at multiple points, and without the need for compressing the surface. An interesting problem is that some screens (notably the Nexus One) can't handle multi-touch well because touches interfere with measurements of one-another. Due to measurements being taken along the axes on the side of the phone, the sensor may sometimes flip the points when two fingers cross the same axis. This type of interaction requirement is rare, but nonetheless if designing applications with this dependency, it will not work as expected on the Nexus One. All touch-screens in general have a problem of precision because the compression is wider than the area of ones fingertip. A study by Holz [1] shows that if point of input perceived by the user is taken into account, improvements of about 1.8 times higher accuracy could be achieved compared to baseline feature sizes. Opinion of minimum size vary, anything around 10mm is borderline and should be implemented with care. Regardless of improvements, any touch-based selection will still need to take variation of difference between the actual and intended selection point into account. Touched-based text-entry for example employs an algorithm to correct for errors when adjacent letters are pressed inadvertently.

Since the principle interaction with the touch-based interface is the point-and-click, apart from differences in accuracy, it is similar to the left-mouse-button and (if lucky) may enable direct translation of some applications. One drawbacks of touch-screens compared to the mouse is that there are no direct equivalent of the mouse right-click or hover, so applications that are dependent on these interactions must be rewritten. For example, a website that uses a drop-down menu depending on a mouse-hover might not be usable on a mobile. Another drawback (compared to the mouse and in general) is addition to the precision mentioned, is the screen being partially obscured by your finger (for instance, anyone who has played Angry Birds will know that fine-tuning the velocity of launch is difficult while your finger is in the way!).

Possible advantages of interfacing with the touch-screen can be enabled by designing for multiple selection points and sliding motions, which opens up a new domain of possible interaction. A combination of sliding motions using one or more finger is called a gesture[1], and these may be used for control without the need for a visual user-interface. Some gestures have become de-facto standards, such as the pinching motion used for zooming control, or horizontal sliding for navigating back- and forward. It will be interesting to see if other gestures will become commoditized in the same way. Some applications are providing enhanced functionality with a variety of gestures[2], such as the Dolphin Browser enables a range of control functions based on gestures[3].

Text-entry, of prevalent important in much of our activities are a significant drawback compared to the conventional keyboard

[1] http://developer.android.com/resources/articles/gestures.html
[2] http://blogs.computerworld.com/16836/android_gestures
[3] http://www.dolphin-browser.com/

(about a typical 25 vs 50wpm), and requires a proportion of the screen for control. Two innovations that improve text rates marginally are Swype and Swiftkey. Swype allows the user to input text by sliding the finger between letters, combined with predictive suggestions for the current word. Swiftkey is based on natural language processing and experience about the particular user's language patterns, while typing it will offer predictions for the current *and* next word. In tests the two techniques have comparable performance advantages, but notably there is no reason why one couldn't combine the two.

Evaluation

The support for accurate multi-touch has only been available for a short time, and only a fraction of the current market is represented. As possibilities of interacting with gestures become more supported, they may replace familiar buttons, menus and tabs, possibly removing visible controls all together in some applications [2].

There are other sensors related to other possible forms of input, it is interesting to consider HCI (Human Computer Interaction) as a whole, further advances may solve some of the inherent problems of mobile I/O. Effective speech-to-text recognition would have the potential to bring the rate of text-entry to that of natural speech (150 wpm).

In terms of application design, the main differences that we have to contend with compared to the desktop are the small screen sizes, significant device-heterogeneity, and a different paradigm of interaction control. This will probably be prevalent challenges for some time to come, but if implemented effectively should bring a large arrays of applications that we've previously associated as desktop activities literally into the palms of our hands.

References

[1] Holz C. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. *Proceedings of the 28th international conference on Human factors in computing systems*, 2010.

[2] Rico J. and Brewster S. Usable gestures for mobile interfaces: evaluating social acceptability. *Proceedings of the 28th international conference on Human factors in computing systems*, 2010.