Lecture 3: Text and gesture interaction

When technical people are commenting on, or even creating, user interfaces, they often get distracted or hung up on the hardware used for input and output. This is a sign that they haven't thought very hard about what is going on underneath and also that they will never keep up with new technical advances. There have always been good and bad examples of interface designs using control panels, punch cards, teletypes, text terminals, bitmap displays, light pens, tablets, mice, touch screens, and so on. With every generation, you can hear people debating whether, for example, 'the mouse is better than a touch screen' or 'voice input is better than a keyboard'. And opinions or expertise on these matters quickly gets out of date. Nowadays, it is more challenging to work with projection surfaces or augmented reality. But sensing and display technologies change fast, and it's more important to understand the principles of interaction than the details of a specific interaction device.

The lecture on visual representation was based on display principles that are independent of any particular display hardware. If we consider the interaction principles that are independent of any particular hardware, these are:

- How does the user get content (both data and structure) into digital form?
- How does the user navigate around the content?
- How does the user manipulate the content (restructuring, revising, replacing)?

General principles: direct manipulation, and interface modes

At the point where the GUI was about to become popular, HCI researcher Ben Shneiderman summarized the important opportunities it provided, under the name ***Direct Manipulation***. In fact some of these things were already possible with text interfaces (for example after the advent of full-screen text editors), and they remain relevant in more recent generations of hardware. The principles of Direct Manipulation as described by Shneiderman are:

 An object that is of interest to the user should be continuously ***visible*** in the form of a graphical representation on the screen

 Operations on objects should involve physical ***actions*** (using a pointing device to manipulate the graphical representation) instead of commands with complex syntax

 The actions that the user makes should be ***rapid***, should offer ***incremental*** changes over the previous situation, and should be ***reversible***

 The ***effect of actions*** should immediately be visible, so that the user knows what has happened

 There should be a modest set of commands doing everything that a novice might need, but it should be possible to expand these, gaining access to more functions as the user develops expertise.

We should also note an additional principle, defined around the same time by Larry Tesler at Apple, that the same action should always have the same effect. Tesler campaigned against 'modes' in the user interface, based on his studies of non-technical users. The largest achievement of the 'windows' style interface is that the frames around each application give the user a clue about different modes – but as Tesler said, removing modes altogether is a great ambition.

Content creation

Text content: including how we can assess and measure the efficiency of alternative text entry mechanisms.

Non-text content: This course won't say very much about non-text content creation. 'Content' can refer to music, visual arts, film, games, novels and many other genres. All of those fields develop their own professional tools, and there is a constant stream of 'amateur' tools modeled on the professional ones. Cultural tastes don't change that fast (the rate of change is generational, not annual), so digital content creation tools are usually derived from and imitate the artistic tools of previous generations (cameras, microphones, mixing desks, typewriters etc). Innovative content creation tools appear first in the 'avant garde' contemporary arts, and take a generation to reach popular

audiences, get taken up by mainstream professional artists, and become subject to consumer demand for amateur tools. The Computer Lab Rainbow Group has always had an active programmer of research engagement with contemporary artists, developing new digital media tools. That research continues actively at present, but is outside the scope of an introduction to HCI.

Content manipulation and navigation via deixis

In order to manipulate content, the user has to be able to refer to specific parts of the product (whether text, diagram, video, audio etc) that he or she is working on. In early text interfaces, references were made by numbering the lines of a text file. As in programming languages, line number could be replaced by labels, but it is irritating to give everything names. Imagine a shop where everything for sale was given its own unique name, or had to be referred to by index position of aisle, shelf, and item. It's much easier just to point and say 'I want that one'. In language, this is called ***deixis*** – sentences in which the object is identified by pointing at it, rather than by naming it. For the same reason, deixis has become universal in computer languages, and this is why devices for pointing are so important in user interfaces. In early GUIs, the combination of mouse and pointer to achieve deixis was a significant invention (hence the WIMP interface – Windows, Icons, Mouse, Pointer). Other inventions around the same time were the placement of a text cursor between characters, rather than identifying a single character. But new hardware suggests new approaches to deixis – touch screens, augmented reality etc will all require new inventions. It's reasonable to assume that deixis in different media can be achieved in different ways too – audio interfaces, cameras, and other devices don't necessarily need to have a cursor. In many cases, what is required is a deictic method that relates user 'gestures' (detected via any kind of sensing device) with a media 'location'. Navigation is then a matter of supporting user strategies to vary that location, including techniques to show local detail within a larger context (via scroll bars, zooming, thumbnails, fisheye views, overview maps, structure navigation and so on) Simple content manipulations include simply adding more content (perhaps inserted

within a particular context), or removing content that isn't required. Anything more complex involves modifying the structure of the content. This is an area in which user interface design can build on insights from the usability of programming languages.

Evaluation of pointing devices and WIMP interfaces

As with text entry, modern user interfaces involve so much pointing, that it is worth optimizing the efficiency of the interaction. Early HCI models based their optimization on ***Fitts' law*** – an experimental observation that the time it takes to point at a given location is related to the size of the target and also the distance from the current hand position to the target.

Fitts original experiment involved two targets of variable size, and separated by a variable distance. Experimental subjects were required to touch first one target, then the other, as quickly as they could. The time that it takes to do this increased with the ***Amplitude*** of the movement (i.e. the distance between the targets) and decreased with the ***Width*** of the target that they were pointing to:

$T = K \log_2(A / W + 1)$ where A = amplitude, W = width

When evaluating new pointing devices, it can be useful experimentally to measure performance over a range of target sizes and motion distances, in order to establish the value of the constant in this equation (sometimes called ID: the ***Index of Difficulty***).

In user interfaces that require a user to make many sequences of repetitive actions (for example, people working in telephone call centres or in data entry), it can be useful to compare alternative designs by counting the individual actions needed to carry out a particular task, including the number and extent of mouse motions, as well as all the keys pressed on the keyboard. This ***Keystroke Level Model*** can be used to provide a quantitative estimate of user performance, and to optimize the design and layout of the interaction sequence. It is more difficult to make numerical comparisons of user interfaces in cases where the user actions are less predictable – the ***GOMS*** model (Goals Operators Methods Selection) combines keystroke-level estimates of user actions with an AI planning model derived from the 1969 work of Ernst and Newell on a ***Generalised***

Problem Solver. The GPS operated in a search space characterised by possible intermediate states between some initial state and a goal state. Problem solving consisted of finding a series of operations that would eventually reach the goal state. This involved recursive application of two heuristics:

A) select an intermediate goal that will ***reduce the difference*** between the current state and the desired state, and

B) if there is no operation to achieve that goal directly, ***decompose it into sub-goals*** until the leaves of the sub-goal hierarchy can be achieved by individual keystrokes or mouse movements.