Human-Computer Interaction

Lecture 3: Text and gesture interaction

VISUAL DESIGN EXERCISE REVIEW

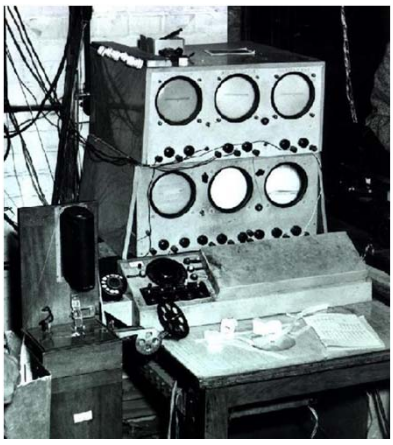
- Visit Guardian data blog
- Browse “Data A-Z”
- Propose new visualisation

	Graphic Resources	Correspondence	Design Uses
Marks	Shape Orientation Size Texture Saturation Colour Line	Literal (visual imitation of physical features) Mapping (quantity, relative scale) Conventional (arbitrary)	Mark position, identify category (shape, texture colour) Indicate direction (orientation, line) Express magnitude (saturation, size, length) Simple symbols and colour codes
Symbols	Geometric elements Letter forms Logos and icons Picture elements Connective elements	Topological (linking) Depictive (pictorial conventions) Figurative (metonym, visual puns) Connotative (professional and cultural association) Acquired (specialist literacies)	Texts and symbolic calculi Diagram elements Branding Visual rhetoric Definition of regions
Regions	Alignment grids Borders and frames Area fills White space Gestalt integration	Containment Separation Framing (composition, photography) Layering	Identifying shared membership Segregating or nesting multiple surface conventions in panels Accommodating labels, captions or legends
Surfaces	The plane Material object on which the marks are imposed (paper, stone) Mounting, orientation and display context Display medium	Literal (map) Euclidean (scale and angle) Metrical (quantitative axes) Juxtaposed or ordered (regions, catalogues) Image-schematic Embodied/situated	Typographic layouts Graphs and charts Relational diagrams Visual interfaces Secondary notations Signs and displays

HISTORIC INTERFACES

Control panels

- The earliest computers were treated like scientific instruments
  - often controlled by physical reconfiguring
  - processes monitored via lamps and CRTs
  - usability considerations are those of machinery



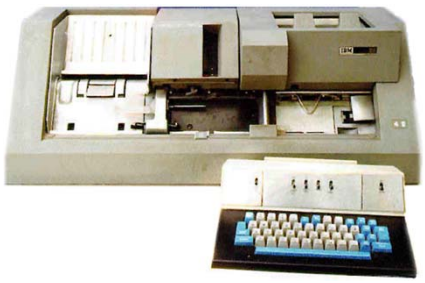
Algebraic languages

- Programmers worked away from the machine creating paper tapes
  - "interface" was of mathematicians working at a desk
  - Automatic Formula Translation (FORTRAN) was a great improvement in usability.

```
    DIMENSION A(11)
    READ A
2   DO 3,8,11 J=1,11
3   I=11-J
    Y=SQRT(ABS(A(I+1)))+5*A(I+1)**3
    IF (400>=Y) 8,4
4   PRINT I,999.
    GOTO 2
8   PRINT I,Y
11  STOP
```

Data files

- Punch-cards were already established for commercial data processing
  - keypunch operators
  - programmers carried boxes of cards
  - interaction consists of filing and paperwork procedures



Command lines

- Teletypes allowed direct interaction, but:
  - local typing often echoed back to paper
  - computer waited to respond after each line
- Command/response, or *dialogue* paradigm

Line editors

- Editing a file via command dialogue
  - user must establish context for commands
  - user must maintain a mental model of current state
  - user's memory constraints require extra actions to confirm state

```
10p
quick brown foz
.s/foz/fox/
+
?
9p
ggggg
.d
```

WYSIWYG

- "Glass teletypes"
  - so-called because no need for paper; instead used "scrolling" up the screen
- The *full-screen* editor
  - control codes allowed cursor positioning
  - allows editing text within the display context
  - user can see the product being worked on
- *"What You See Is What You Get"*

Modeless interaction

- Early full-screen editors (e.g. vi) were like previewer front ends for line editors.
  - Commands issued from separate command line
  - *Modal* interaction confusing and unpredictable
- In *modeless* editors (e.g. emacs)
  - Given keystroke has the same effect in any context

Menus

- What commands can I perform (line editor)?

```
:afb21$ ex
Entering Ex mode.  Type "visual" to get out.
:help
"help.txt" [readonly] 1185 lines,
55790 characters
:
```

- Better to list them, allowing the user to choose
  - menu screen, or “pop-up” if a full screen available

Pointing devices

- How to select from a full screen menu without many cursor movements?
  - Tab to position
  - Light pens
    - point directly at a place on the screen
  - Joystick
    - directional motion
  - Mouse
    - near keyboard (unlike light pen), while giving positional control (unlike joystick)

Graphical displays

- Originally for technical applications:
  - draughting
  - schematics
- Allowed preview of graphic operations
  - Often toggled between text (control) mode and graphic (output) mode.

Icons and windows

- Largely developed at Xerox Palo Alto Research Centre (PARC)
  - Star and Alto projects
  - *bitmapped* displays
- Multiple contexts shown by frames.
- Pictures used to represent abstract entities.

DIRECT MANIPULATION

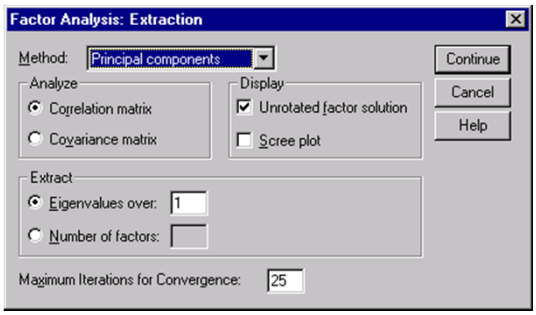
Direct manipulation

- WIMP: **w**indow / **i**con / **m**enu / **p**ointer
- Rapidly being superseded by touch devices
- Most radical change in WIMP:
  - Command is not unit of interaction
  - Object of interest is unit of interaction (Sutherland, Smith)

Direct manipulation

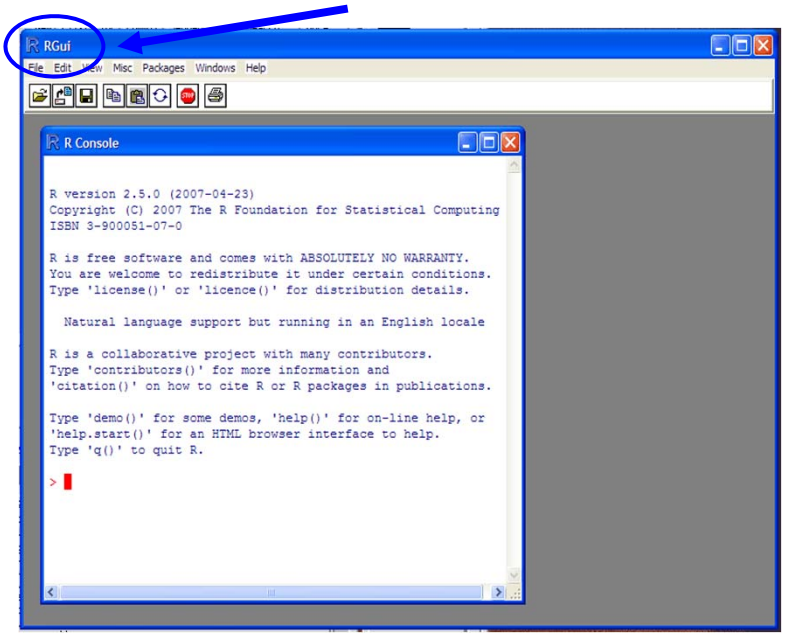
- Described by Shneiderman:
  - objects of interest continuously *visible*
  - operations by physical *actions*, not commands
  - actions *rapid*, *incremental*, *reversible*
  - *effect* of actions immediately visible
  - basic commands for novices, more for experts
- Unfortunately these things are not true of all "graphical user interfaces".

Graphics without directness



- Object of analysis isn't visible
- Effect of controls isn't visible
- All functions are presented to novices

Even less direct!



Fitts' law

- Fitts' law models the speed-accuracy trade-off in pointing

$$T = a + bID$$

$$ID = \log_2\left(\frac{D+W}{W}\right)$$

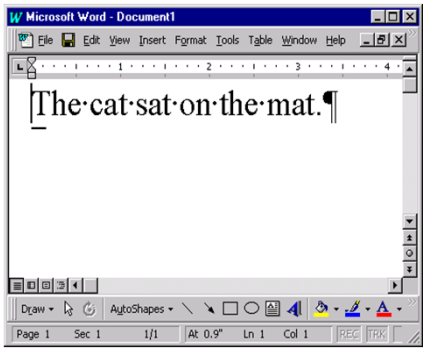
- A model allows prediction and optimisation

GOMS: Goals, Operators, Methods, Selection

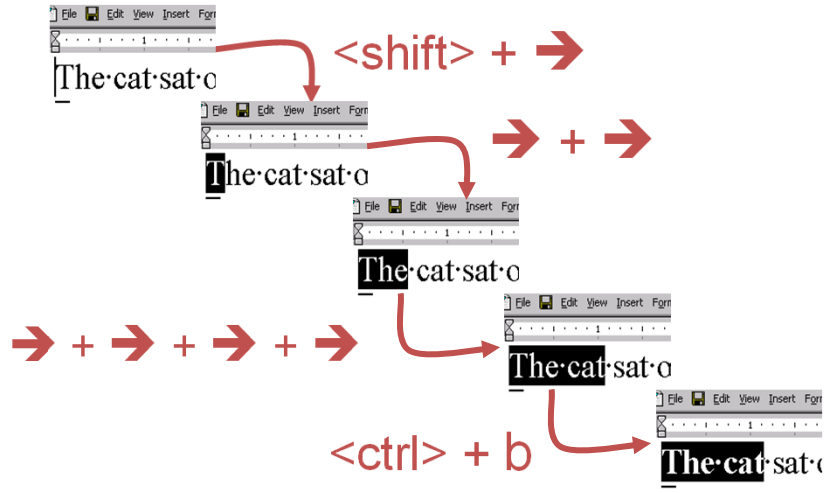
- Goals: what is the user trying to do?
- Operators: what actions must they take?
  - **H**ome hands on keyboard or mouse
  - **K**ey press & release (tapping keyboard or mouse button)
  - **P**oint using mouse/lightpen etc
- Methods: what have they learned in the past?
- Selection: how will they choose what to do?
  - **M**ental preparation

Aim is to predict speed of interaction

- Which is faster? Make a word bold using
  a) Keys only
  b) Font dialog



Keys-only method



Keys-only method

- Mental preparation: **M**
- Home on keyboard: **H**
- Mental preparation: **M**
- Hold down shift: **K**
- Press → : **K**
- Press → : **K**
- Press → : **K**
- Press → : **K**
- Press → : **K**
- Press → : **K**
- Press → : **K**
- Release shift: **K**
- Mental preparation: **M**
- Hold down control: **K**
- Press b: **K**
- Release control: **K**

Keys-only method

- 1 occurrence of H — 0.40
- 3 occurrences of M — 1.35 * 3
- 12 occurrences of K — 0.28 * 12

7.81 seconds

Font dialog method

Motion times from Fitts' law

- From start of "The" to end of "cat" ($t \sim k \log (A/W)$):
  - distance 110 pixels, target width 26 pixels, t = 0.88 s
- From end of "cat" to Format item on menu bar:
  - distance 97 pixels, target width 25 pixels, t = 0.85 s
- Down to the Font item on the Format menu:
  - distance 23 pixels, target width 26 pixels, t = 0.34 s
- To the "bold" entry in the font dialog:
  - distance 268 pixels, target width 16 pixels, t = 1.53 s
- From "bold" to the OK button in the font dialog:
  - distance 305 pixels, target width 20 pixels, t = 1.49 s

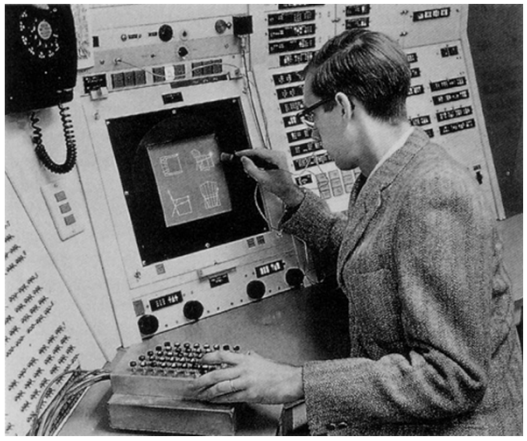
Font dialog method

- Mental preparation: **M**
- Reach for mouse: **H**
- Point to "The": **P**
- Click: **K**
- Drag past "cat": **P**
- Release: **K**
- Mental preparation: **M**
- Point to menu bar: **P**
- Click: **K**
- Drag to "Font": **P**
- Release: **K**
- Mental preparation: **M**
- Move to "bold": **P**
- Click: **K**
- Release: **K**
- Mental preparation: **M**
- Move to "OK": **P**
- Click: **K**

Font dialog method

• 1 occurrence of H	0.40
• 4 occurrences of M	1.35 * 4
• 7 occurrences of K	0.28 * 7
• 6 mouse motions P	1.1 + 0.88 + 0.85 + 0.34 + 1.53 + 1.49
• Total for dialog method:	**13.95 seconds**
	vs.
• Total for keyboard method:	**7.81 seconds**

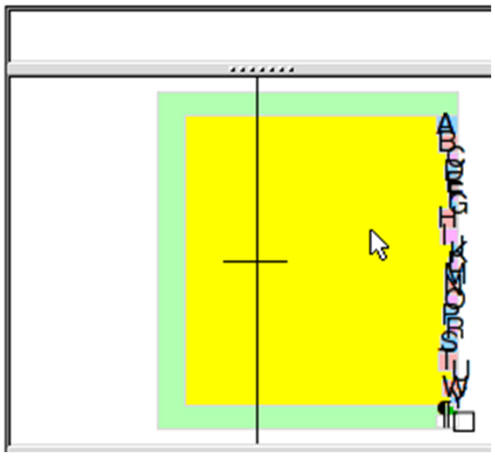
GESTURES



Take another look …

Palm Pilot – 1996

Dasher – 2000



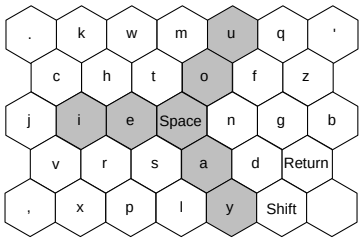
“BREAKING” FITTS’ LAW

$$T \propto \frac{D}{W}$$

- Minimize D – minimize the average distance the pen or finger travels across the keyboard
- Maximize W – maximize the size of the user’s currently intended key (impossible?)

Minimising D with optimized keyboard layouts

Q	F	U	M	C	K	Z
		O	T	H		
B	S	R	E	A	W	X
		I	N	D		
J	P	V	G	L	Y	F1



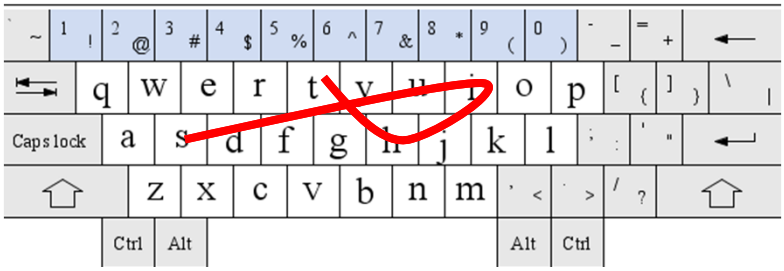
The design context (Per Ola Kristensson)

- Minimize two interdependent variables:
  - Time: mean time for users to achieve the task
  - Error: mean error rate during the task
- Other design considerations:
  - Is this technique easy-to-learn?
  - Is this technique designed for all users or a subset of them?
  - Does this technique rely on expensive high-quality sensors?
  - Is this technique fun to use or terribly tedious or boring?
  - Does this technique depend on other expensive resources (such as high-quality language models)?
  - Is this technique easy to integrate with the rest of the product system?

Can we do better?

- Optimized keyboards are hard to learn
- …and still bound by an inherent Fitts' law speed-accuracy tradeoff
- …and users find them tedious

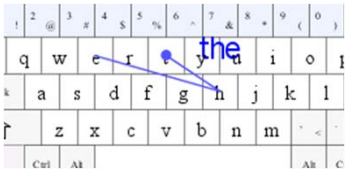
Case study: the "gesture keyboard"

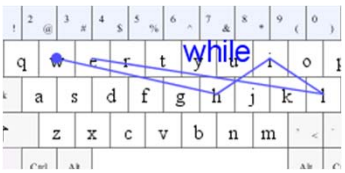


The ShapeWriter gesture keyboard

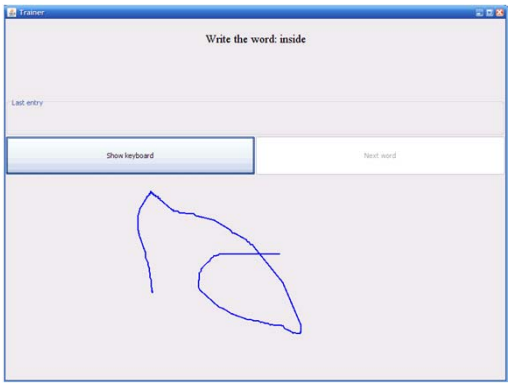
- The vast majority of key combinations do not form valid words
- Let's collect the once who do: into a large lexicon (> 60,000 words)
- Then create shapes on the keyboard for all words in the lexicon

Words form patterns on the layout

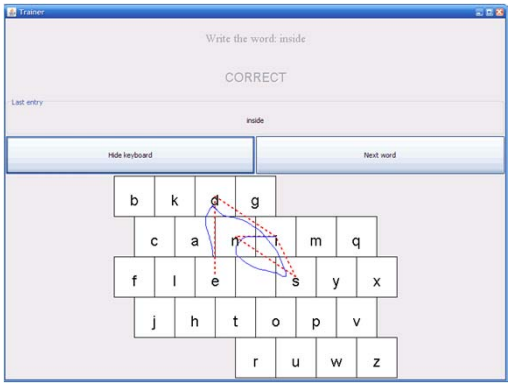




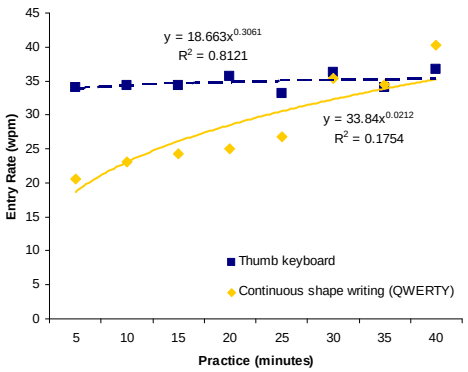
Do users actually reach automaticity?



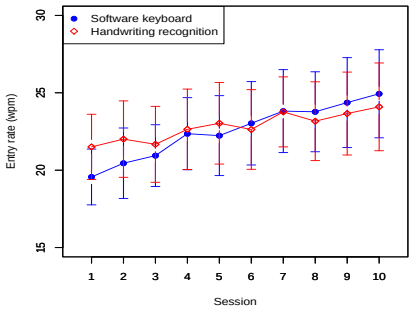
Do users actually reach automaticity?

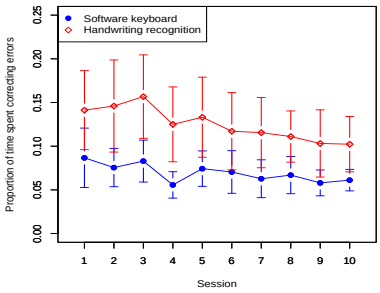


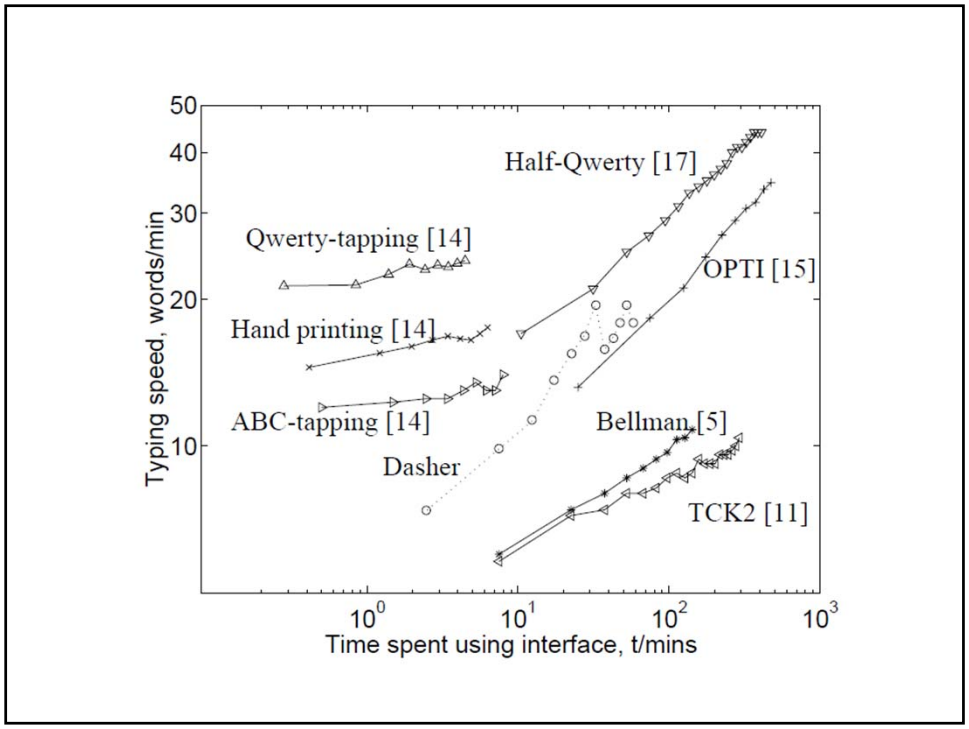
Do novice users get it?



Speed and accuracy







SPEECH, ERRORS AND DEIXIS

(videos, if time)

http://www.inference.phy.cam.ac.uk/kv227/videos/