Text Entry Overview

Prof. Wobbrock

Winter 2018

Introducing text entry

- Text is everywhere
- Even right here
- It had to get here
- It had to be entered
- That's text entry ☺

Text entry's many features

- Letters, cases
- Numbers
- Symbols
- Correction
- Cursor control
- …
- Recognition
- Completion
- Prediction
- Visualization
- Display, feedback
- …

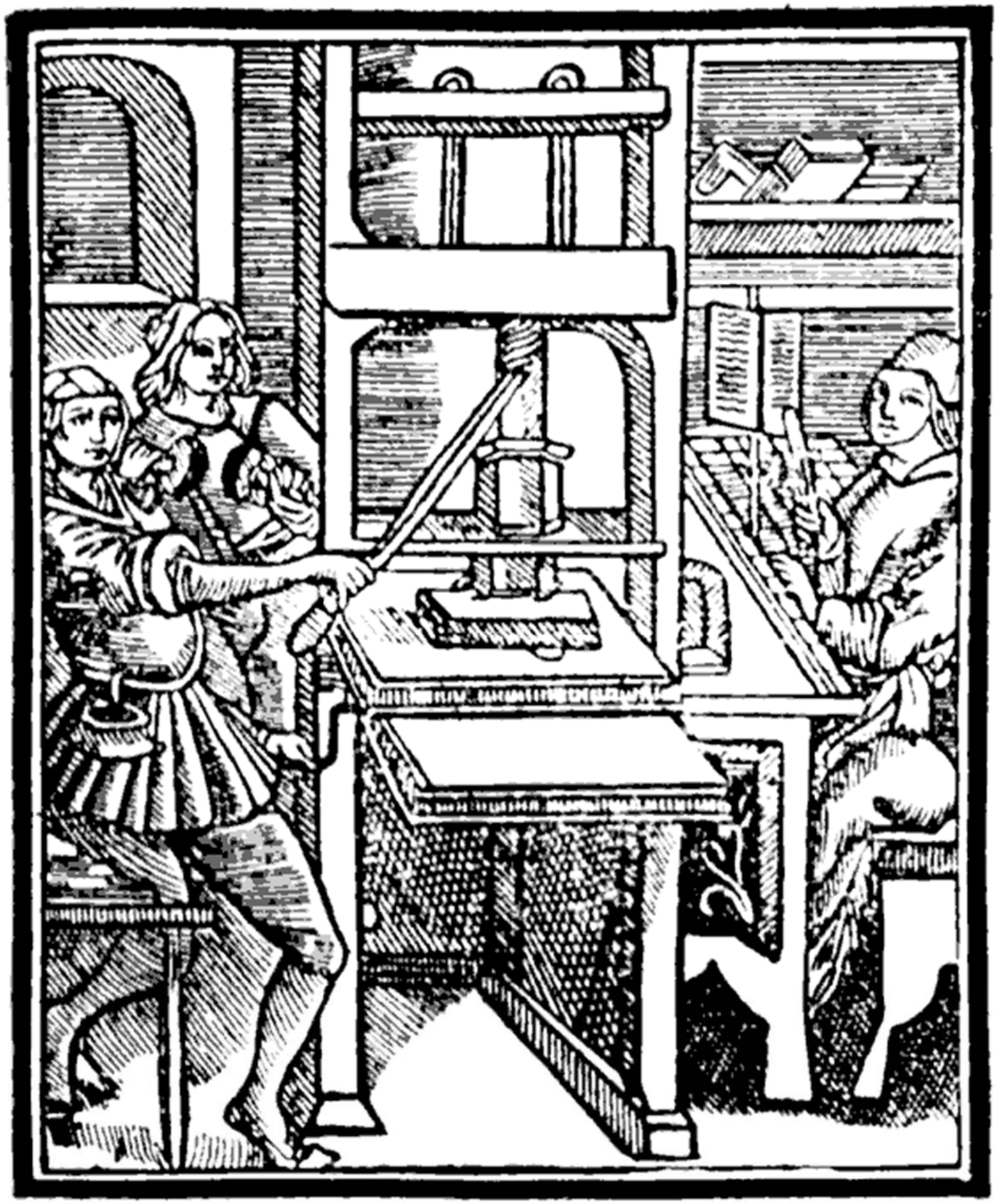
Challenges

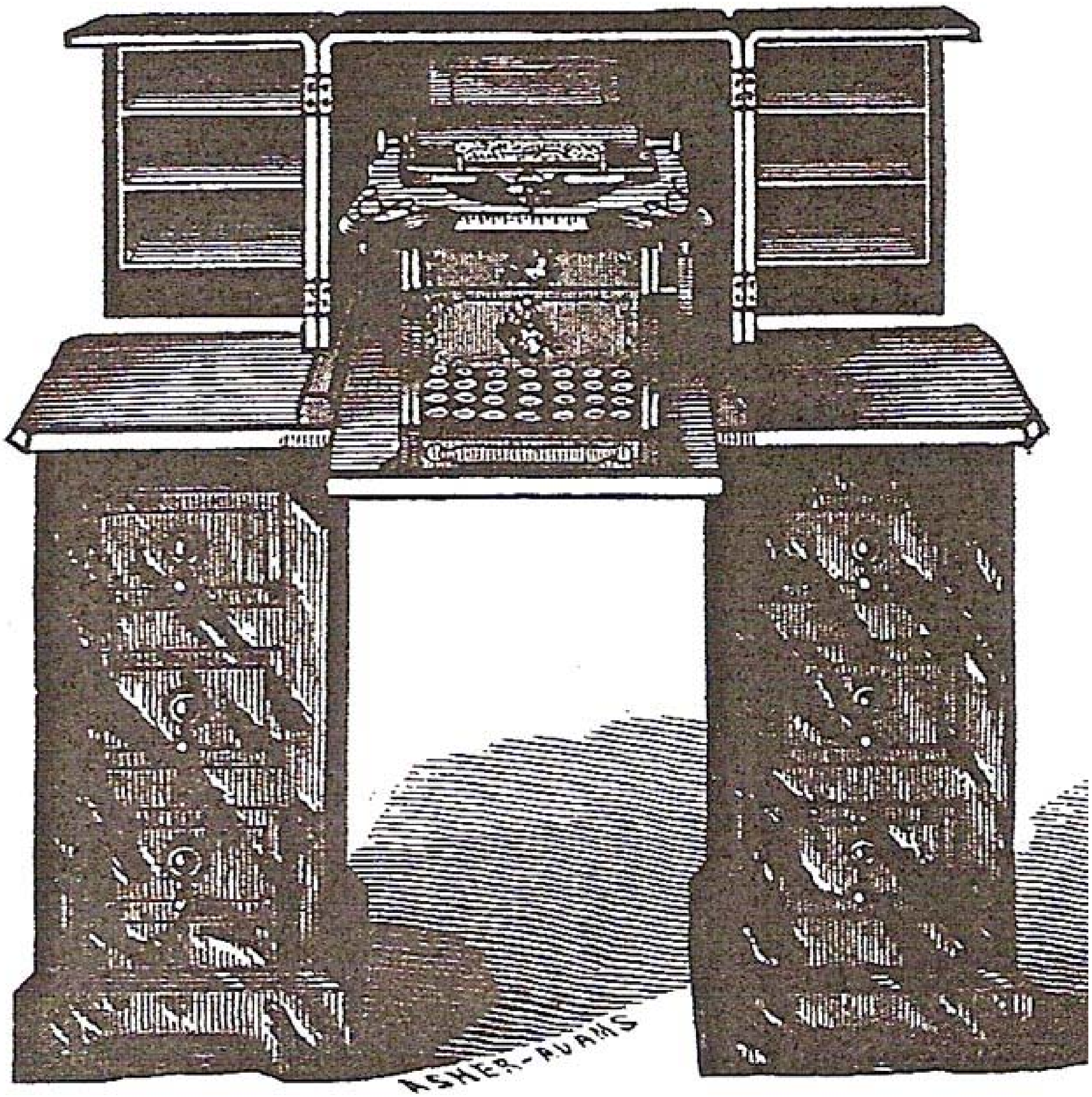
- A writer has to be
  - Fast
  - Accurate
  - In control
  - Efficient
  - Comfortable

- On mobile devices, a writer must do all this in a very small space.

Some entry rates (wpm)

- Hand printing
  - 10-20
- Cursive handwriting
  - 25-35
- Palm OS® Graffiti
  - 15-25
- On-screen keyboards
  - 10
- Stylus keyboard
  - 15-25
- Morse code
  - 25-30
- QWERTY hunt-and-peck
  - 25-35
- QWERTY touch-typing
  - 50-90
- Stenography (shorthand)
  - 100-250
- Speaking
  - 150-220
- Court reporter
  - 225
- Reading
  - 200-300
- Thinking
  - ??? ☺





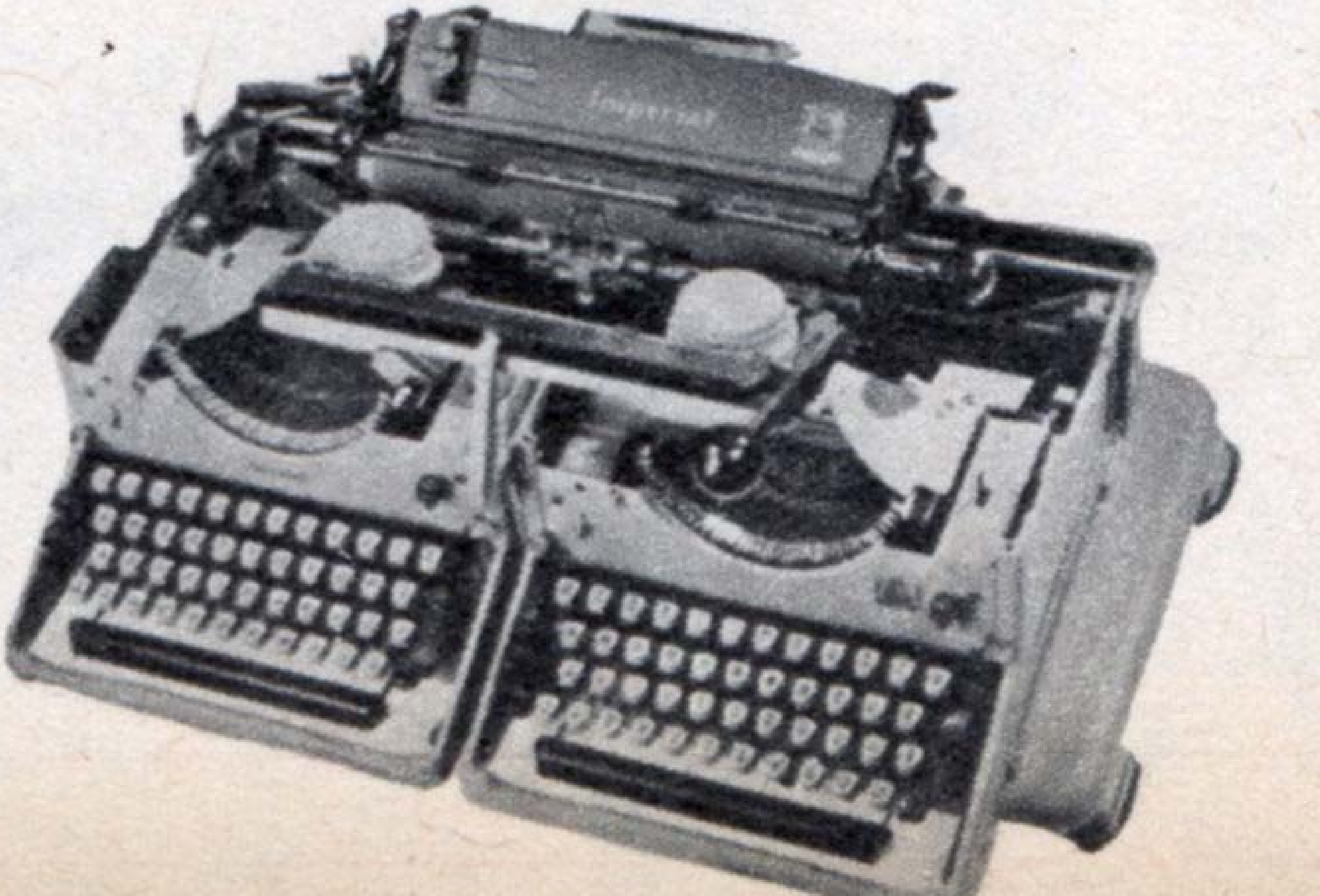
http://www.theoriginof.com/typewriter.html

Sholes–Glidden typewriter.

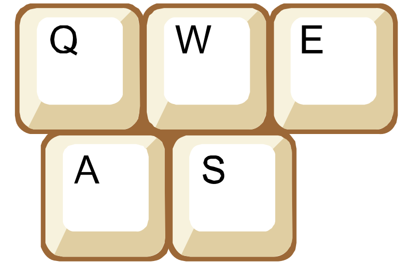

Jewett Typewriter Co
Des Moines, Iowa, U.S.A.
Jewett No. 4


Smith Premier
SMITH PREMIER TYPEWRITER

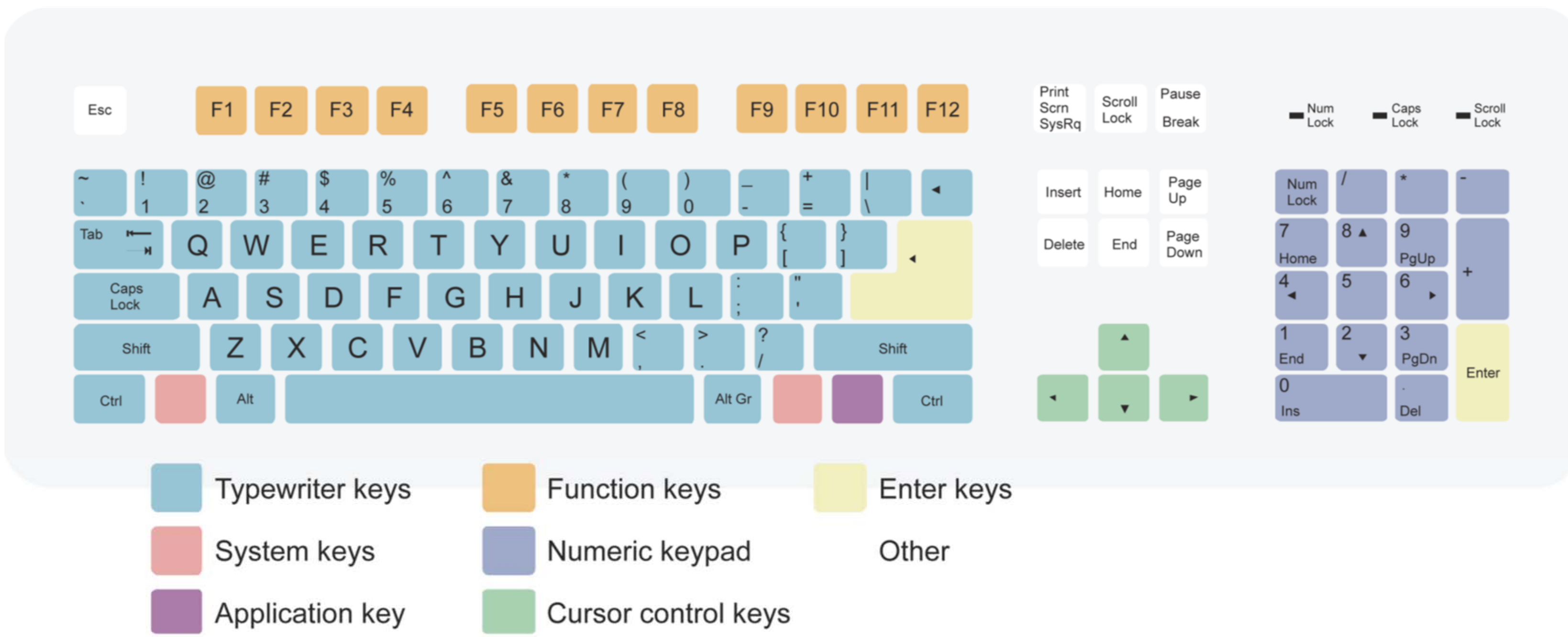
DOUBLE KEYBOARD machine types either plain talk or scientific equations. Imperial Typewriter Company, Leicester, England.



QWERTY (Sholes 1860s)

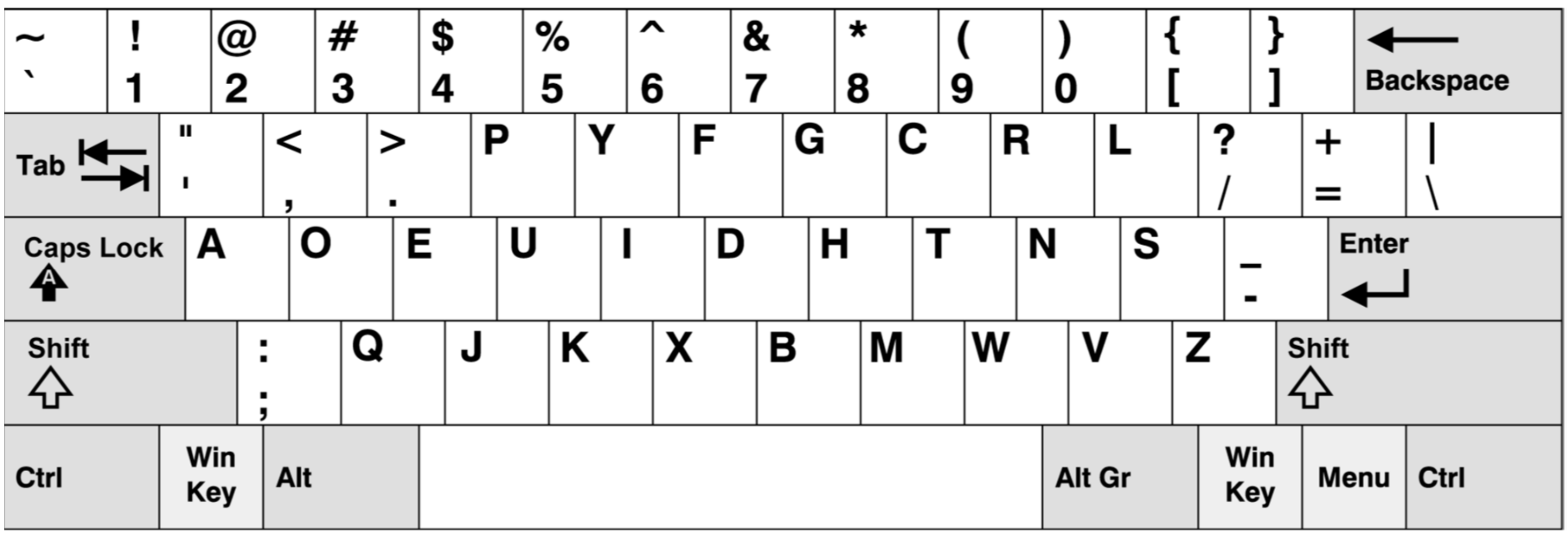


- What was the design principle?
  - Minimize mechanical jamming by alternating hands



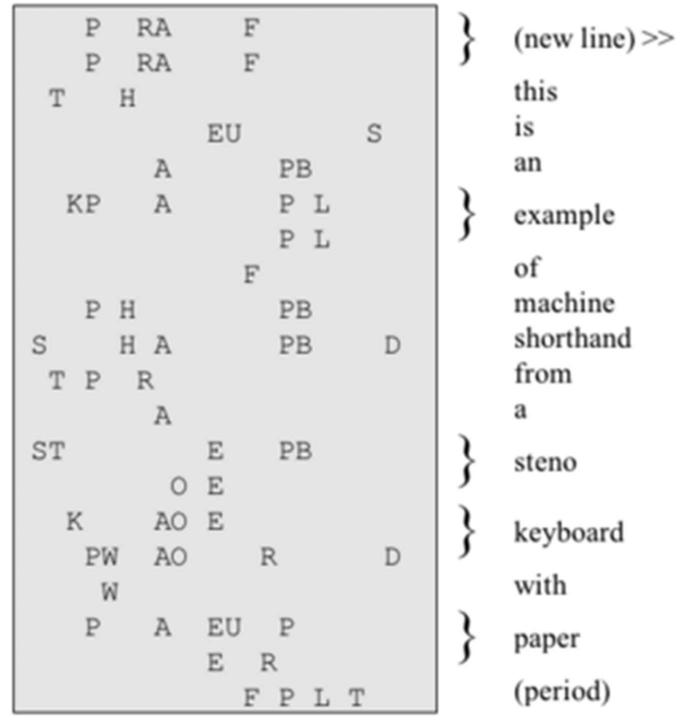
Dvorak (Dvorak and Dealey 1936)

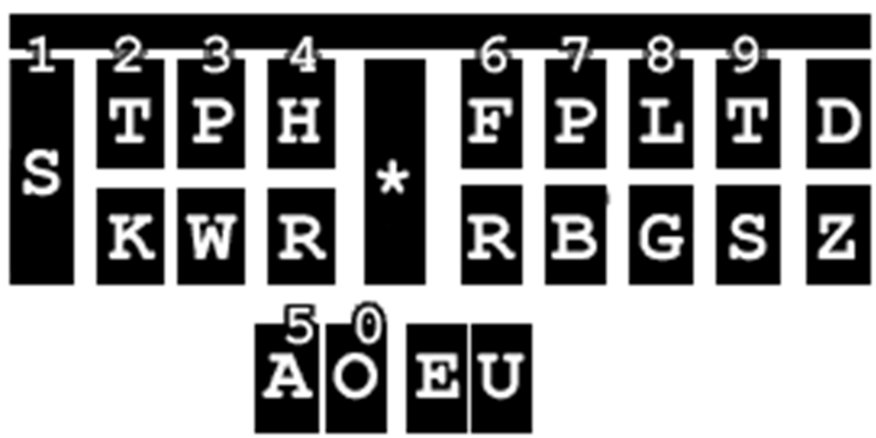
- Alternate layout designed to place most common letters in the home row and to maximize alternation between hands.



Stenotype (Stone 1913)

- Chording keyboard
- Phonetic, personalized
- 2-4 years of training to reach 225 wpm





Morse code (Morse and Vail 1840s)

- Series of dots, dashes, and pauses
  - Not technically binary
  - 25-30 wpm

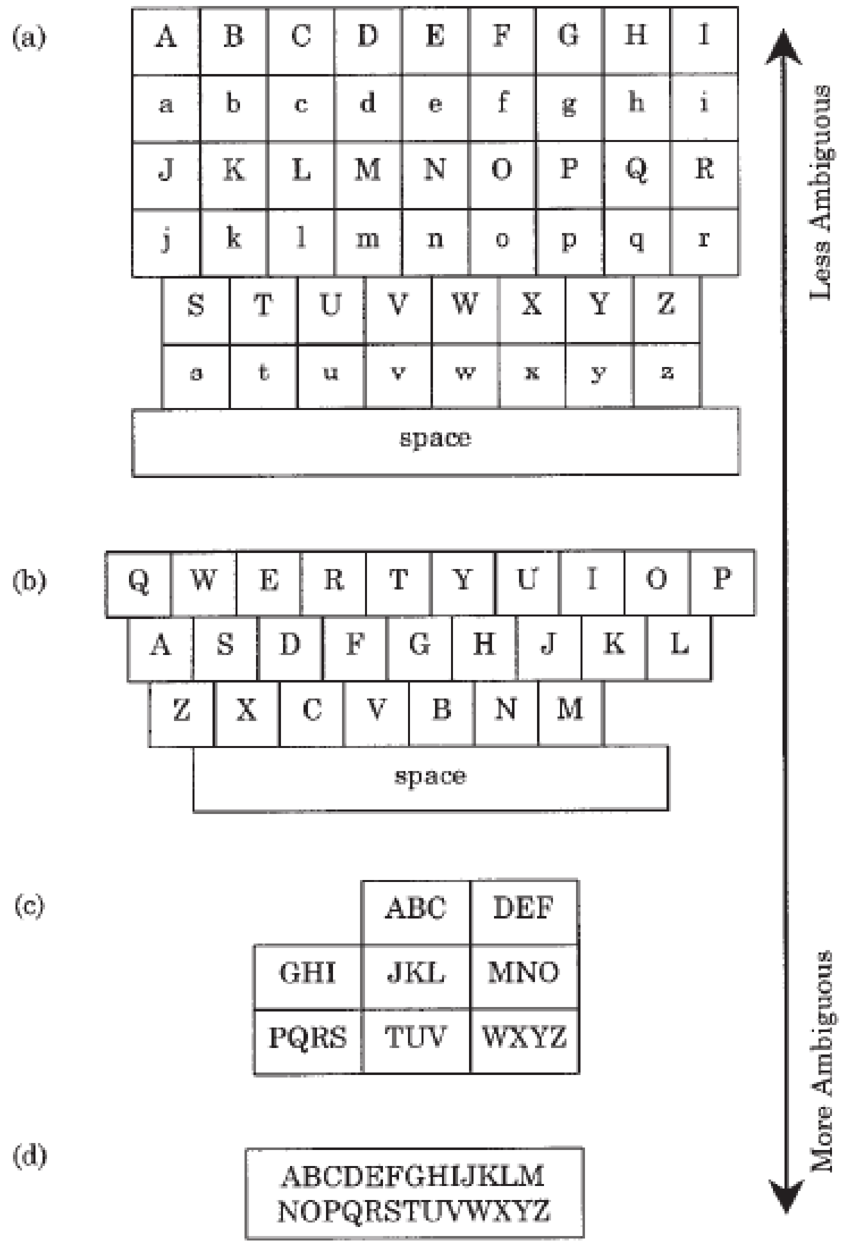
```
-- --- .-. ... .    -.-. --- -.. .
M   O   R   S  E    C    O   D  E
```

http://www.ebaumsworld.com/video/watch/80519289/

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A	• —	U	• • —
B	— • • •	V	• • • —
C	— • — •	W	• — —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — •		
H	• • • •		
I	• •		
J	• — — —		
K	— • —	1	• — — — —
L	• — • •	2	• • — — —
M	— —	3	• • • — —
N	— •	4	• • • • —
O	— — —	5	• • • • •
P	• — — •	6	— • • • •
Q	— — • —	7	— — • • •
R	• — •	8	— — — • •
S	• • •	9	— — — — •
T	—	0	— — — — —


(a)
A B C D E F G H I
a b c d e f g h i
J K L M N O P Q R
j k l m n o p q r
S T U V W X Y Z
s t u v w x y z
space
(b)
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
space
(c)
ABC DEF
GHI JKL MNO
PQRS TUV WXYZ
(d)
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
Less Ambiguous
More Ambiguous

3-key (date stamp)

- Linear 1D keyboard
- Keys: left, right, and select
- Sometimes a thumbwheel, scroll wheel, or joystick instead (iPod uses this)
- Experts: ~9 wpm

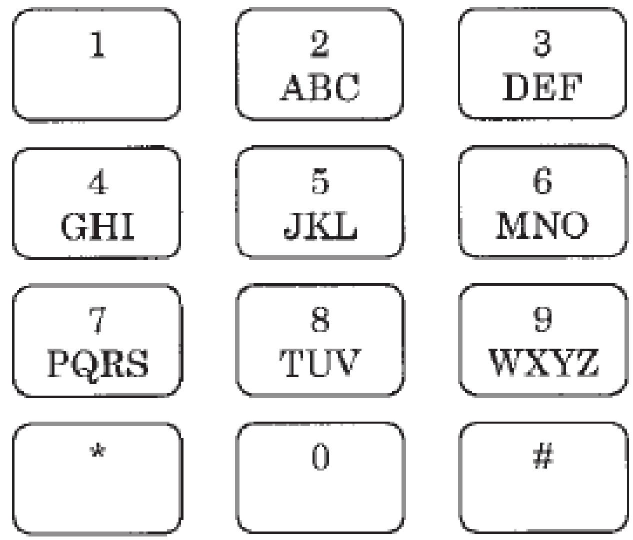
5-key (selection keyboard)

- Spatial 2D keyboard
- Keys: left, right, up, down, select
- Used on some two-way pagers
- Experts: ~10 wpm

a	b	c	d	e	f	g	h	i	j	
k	l	m	n	_	o	p	q	r		
			s	t	u	v	w	x	y	z

Multitap

- Telephone keypad method
- Press each key the number of times corresponding to the position of the desired letter
- For successive letters on same key, use a 1.5 timeout, or NEXT key
- 2.0342 KSPC
- About 9-10 wpm

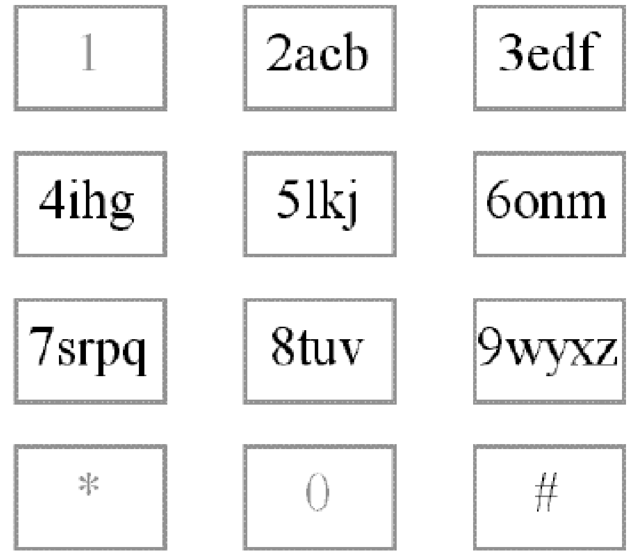


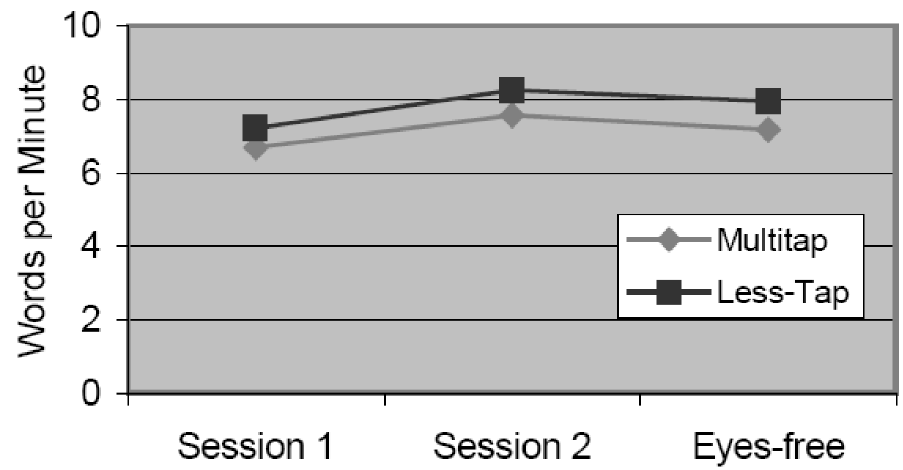
36664 ⏲ 4999

?

Less-Tap (Pavlovych & Sturzlinger 2003)

- Same as Multitap, but letters arranged by frequency within keys
- 1.5266 KSPC
  - 25% better than Multitap
- About 9-10 wpm





T9 (www.tegic.com)

- Disambiguation-based method for mobile phones
- Uses lexicon with word frequencies to match key sequences to most likely word (only one keypress per letter is made)
- ~15 WPM
- What are some challenges?

http://en.wikipedia.org/wiki/T9_%28predictive_text%29

T9 challenges

843	78425	27696	369	58677	6837	843	5299	364
the	**quick**	**brown**	**fox**	**jumps**	**over**	**the**	jazz	**dog**
tie	stick	crown		lumps	muds	tie	**lazy**	fog
vie						vie		

```
able 2-2-5-3-0
cake 2-2-5-3-N-0
bald 2-2-5-3-N-N-0
calf 2-2-5-3-N-N-N-0
```

Mini-QWERTY

- Thumb-driven QWERTY keyboard
- ~1 KSPC
- 31 WPM novice, 60 WPM expert
- “BlackBerry thumb” can be a problem

BlackBerry Q10

http://en.wikipedia.org/wiki/BlackBerry_thumb

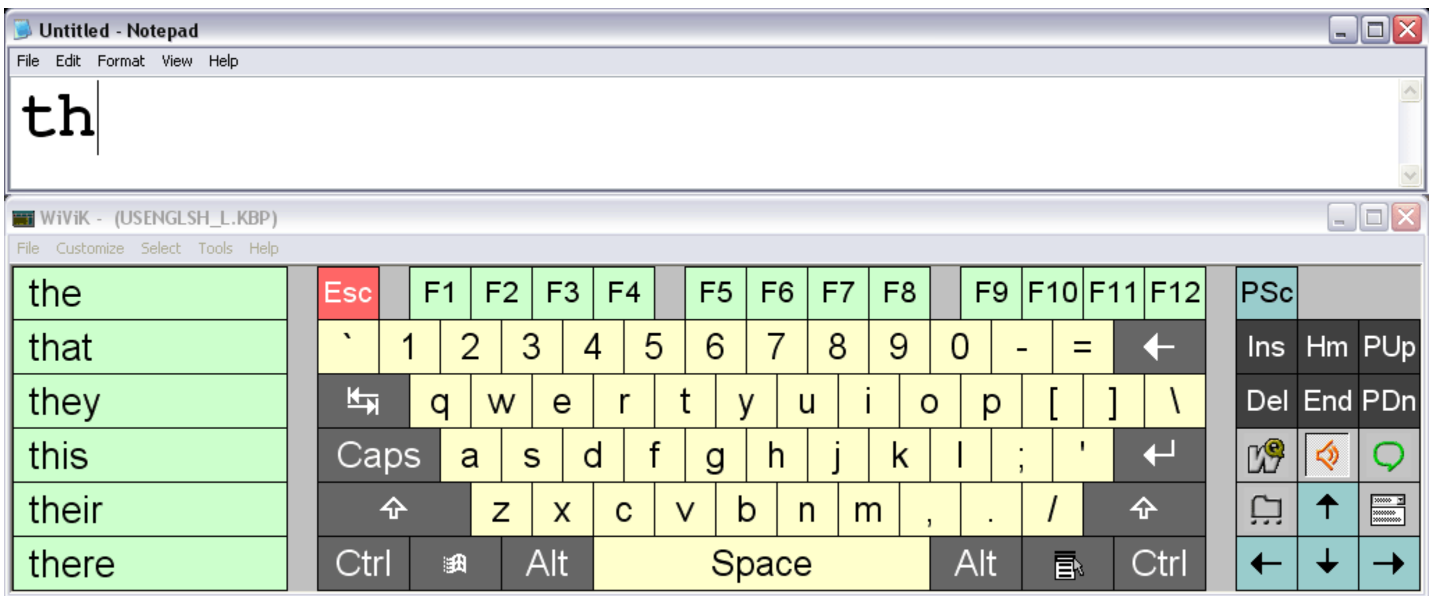
Touch-screen mini-QWERTY

- Virtual mini-QWERTY
- No haptic feedback
- Can automatically adjust key regions based on letter likelihood
  - No visual change
- http://www.youtube.com/watch?v=a-9UggQV9BM

On-screen keyboards

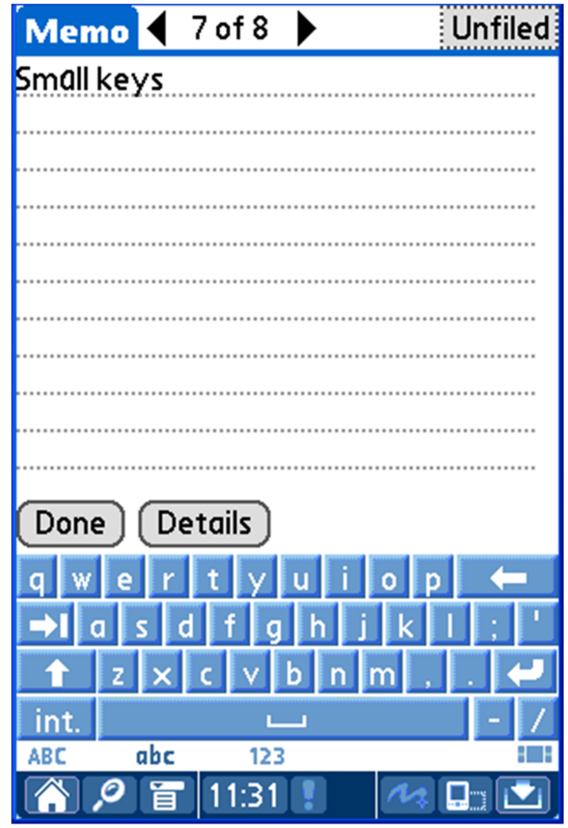
- Move a key selector (like in 5-key) or use a mouse or other pointing device.
- Often used in assistive technology (e.g., with a trackball and dwell).

(WiViK keyboard)



Stylus keyboards

- Use a stylus to tap between keys
- Fitts' law has been used to model (evaluate) and to optimize (generate)
- What other information besides Fitts' law do we need to do this?



Letter digraphs

First Letter	Second Letter: A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Space	Total
A	2	144	308	382	1	67	138	9	322	7	146	664	177	1576	1	100	-	802	683	785	87	233	57	14	319	12	**50**	7086
B	136	14	-	-	415	-	-	-	78	18	-	98	1	-	240	-	-	88	15	7	256	1	1	-	13	-	**36**	1417
C	368	-	13	-	285	-	-	412	67	-	178	108	-	1	298	-	1	71	7	154	34	-	-	-	9	-	**47**	2053
D	106	1	-	37	375	3	19	-	148	1	-	22	1	2	137	-	-	83	95	3	52	5	2	-	51	-	**2627**	3770
E	670	8	181	767	470	103	46	15	127	1	35	332	187	799	44	90	9	1314	630	316	8	172	106	87	189	2	**4904**	11612
F	145	-	-	-	154	86	-	-	205	-	-	69	3	-	429	-	-	188	4	102	62	-	-	-	4	-	**110**	1561
G	94	1	-	-	289	-	19	288	96	-	-	55	1	31	135	-	-	98	42	6	57	-	1	-	2	-	**686**	1901
H	1164	-	-	-	3155	-	-	1	824	-	-	5	1	-	487	2	-	91	8	165	75	-	8	-	32	-	**715**	6733
I	23	7	304	260	189	56	233	-	1	-	86	324	255	1110	88	42	2	272	484	558	5	165	-	15	-	18	**4**	4501
J	2	-	-	-	31	-	-	-	9	-	-	-	-	-	41	-	-	-	-	-	56	-	-	-	-	-	**-**	139
K	2	-	-	-	337	-	-	-	127	-	-	10	1	82	3	1	-	-	50	-	3	-	-	-	8	-	**309**	933
L	332	4	6	289	591	59	7	-	390	-	38	546	30	1	344	34	-	11	121	74	81	17	19	-	276	-	**630**	3900
M	394	50	-	-	530	6	-	-	165	-	-	4	28	4	289	77	-	-	53	2	85	-	-	-	19	-	**454**	2160
N	100	2	98	1213	512	5	771	5	135	8	63	80	-	54	349	-	3	2	148	378	49	3	2	2	115	-	**1152**	5249
O	65	67	61	119	34	80	9	1	88	3	123	218	417	598	336	138	-	812	195	415	1115	136	398	2	47	5	**294**	5776
P	142	-	1	-	280	1	-	24	97	-	-	169	-	-	149	64	-	110	48	40	68	-	3	-	14	-	**127**	1337
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	66	-	-	-	-	-	**-**	66
R	289	10	22	133	1139	13	59	21	309	-	53	71	65	106	504	9	-	69	318	190	89	22	5	-	145	-	**1483**	5124
S	196	9	47	-	626	-	1	328	214	-	57	48	31	16	213	107	8	-	168	754	175	-	32	-	34	-	**2228**	5292
T	259	2	31	1	583	1	2	3774	252	-	-	75	1	2	331	-	-	187	209	154	132	-	84	-	121	1	**2343**	8545
U	45	53	114	48	71	10	148	-	65	-	-	247	87	278	3	49	1	402	299	492	-	-	-	1	7	3	**255**	2678
V	27	-	-	-	683	-	-	-	109	-	-	-	-	-	33	-	-	-	-	-	1	-	-	-	11	-	**-**	864
W	595	3	-	6	285	-	-	472	374	-	1	12	-	103	264	-	-	35	21	4	2	-	-	-	-	-	**326**	2503
X	17	-	9	-	9	-	-	-	10	-	-	-	-	-	1	22	-	-	-	23	8	-	-	-	-	-	**21**	120
Y	11	10	-	-	152	-	1	1	32	-	-	7	1	-	339	16	-	-	81	2	1	-	2	-	-	-	**1171**	1827
Z	3	-	-	-	26	-	-	-	2	-	-	4	-	-	2	-	-	-	3	-	-	-	-	-	3	9	**2**	54
Space	**1882**	**1033**	**864**	**515**	**423**	**1059**	**453**	**1388**	**237**	**93**	**152**	**717**	**876**	**478**	**721**	**588**	**42**	**494**	**1596**	**3912**	**134**	**116**	**1787**	**-**	**436**	**2**	**-**	**19998**
Total	7069	1418	2059	3770	11645	1549	1906	6739	4483	131	932	3885	2163	5241	5781	1339	66	5129	5278	8536	2701	870	2507	121	1855	52	**19974**	107199

Figure 1. 27 × 27 digrams for the text-entry task. The core 26 x 26 digrams are from Mayzner and Tresselt's (1965) table 2. The space digrams (shaded) were compiled from Mayzner and Tresselt's frequency counts for start-of-word and end-of-word digrams.

(Soukoreff & MacKenzie 1995)

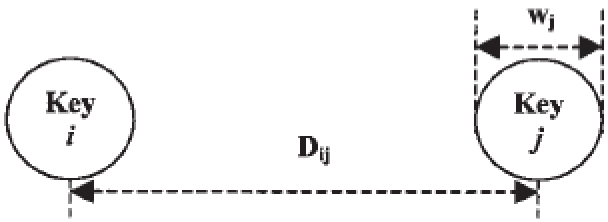
OPTI II (MacKenzie & Zhang 1999)

- Placed 10 most common keys in center
- Then added most common digraphs
- Used trial-and-error
- Fitts' law predicts about 36 wpm

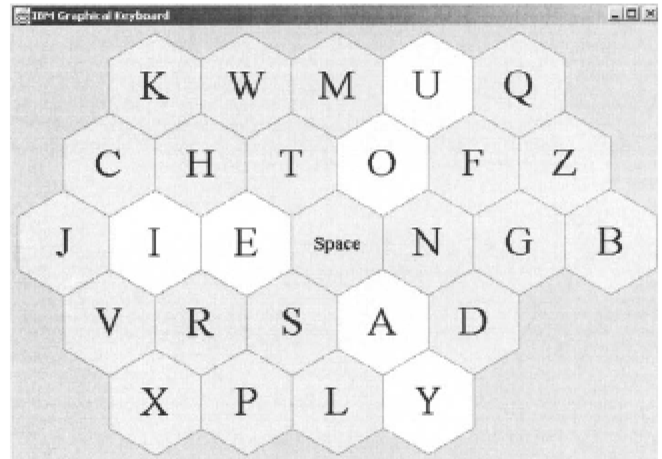
Q	K	C	G	V	J
	S	I	N	D	
W	T	H	E	A	M
	U	O	R	L	
Z	B	F	Y	P	X

(OPTI II from Zhai et al. 2002)

Metropolis keyboards (Zhai et al. 2000)



$$MT = a + b\log_2\left(\frac{D_{ij}}{W_j} + 1\right),$$



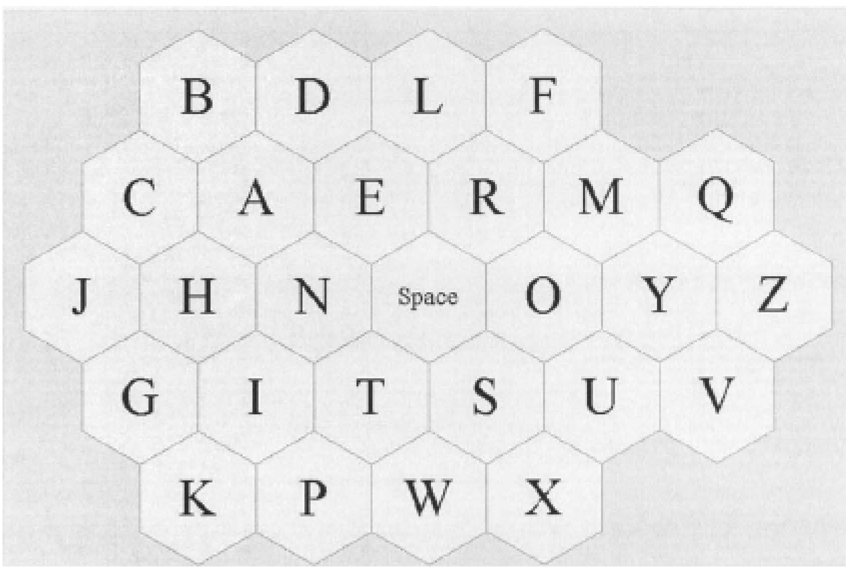
If the frequency of letter j to follow letter i (digraph I–j) among all digraphs is P_{ij}, then the mean time in seconds for typing a character is:

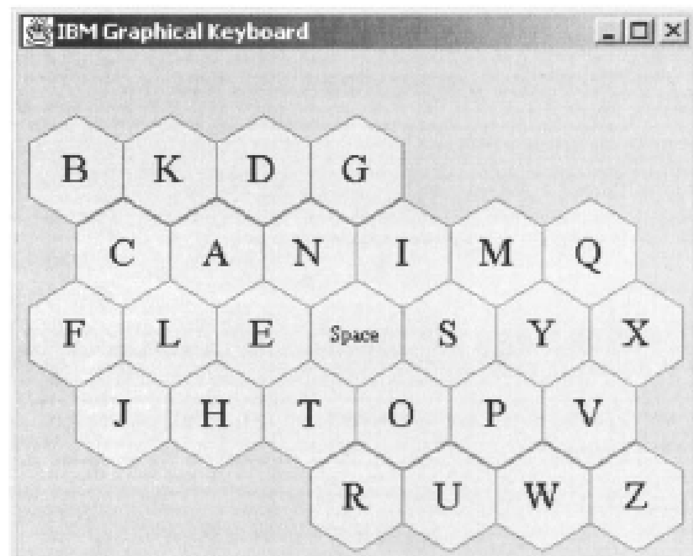
$$t = \sum_{i=1}^{27}\sum_{j=1}^{27}\frac{P_{ij}}{IP}\left[log_2\left(\frac{D_{ij}}{W_j} + 1\right)\right], \tag{2}$$

Assuming five characters per word (including space key), this equation allows us to calculate tapping speed in wpm (60 / 5 t).

ATOMIK (Zhai et al. 2002)

- "Alphabetically tuned and optimized mobile interface keyboard."

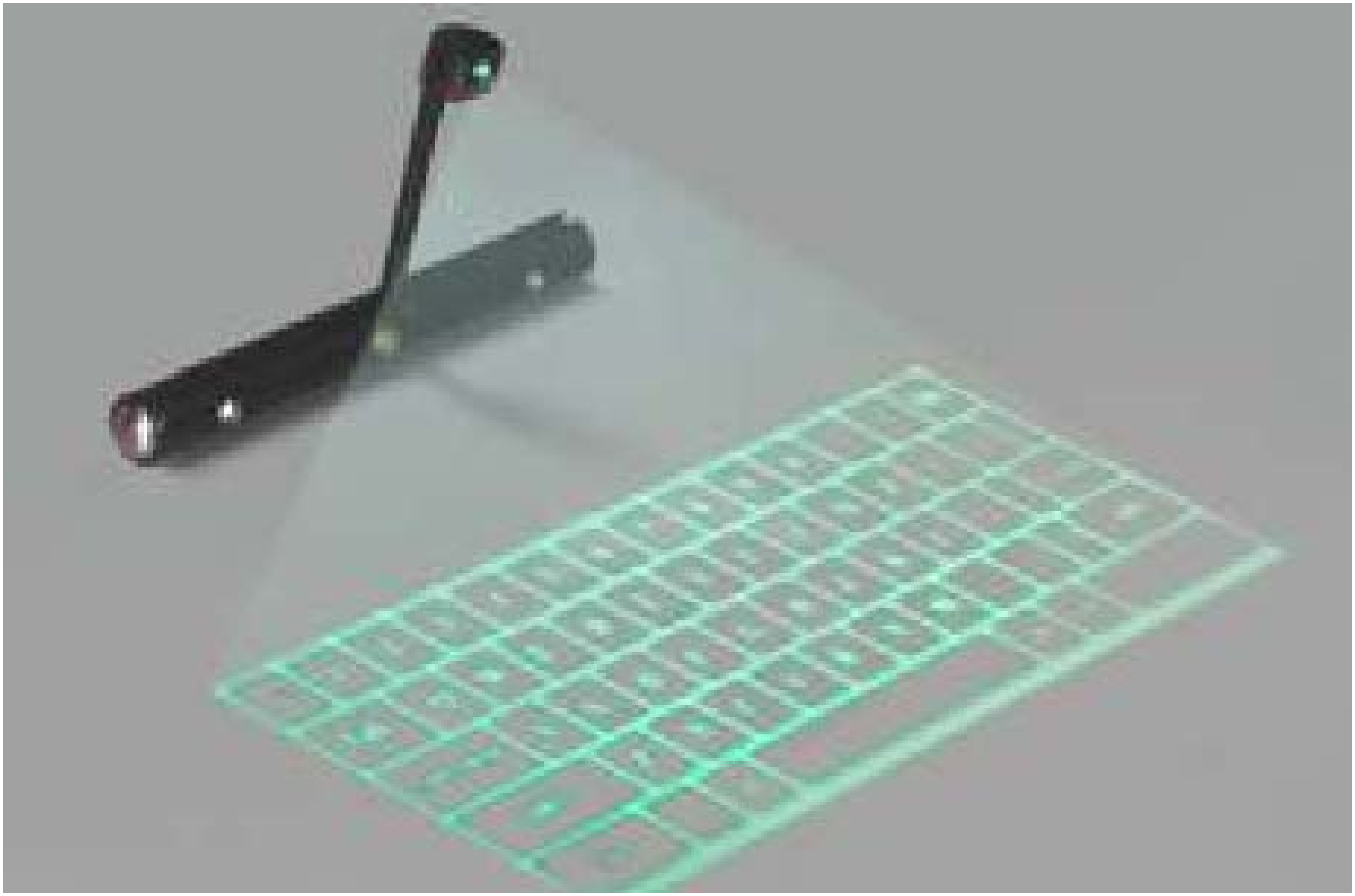




~42 wpm as modeled by Fitts' law

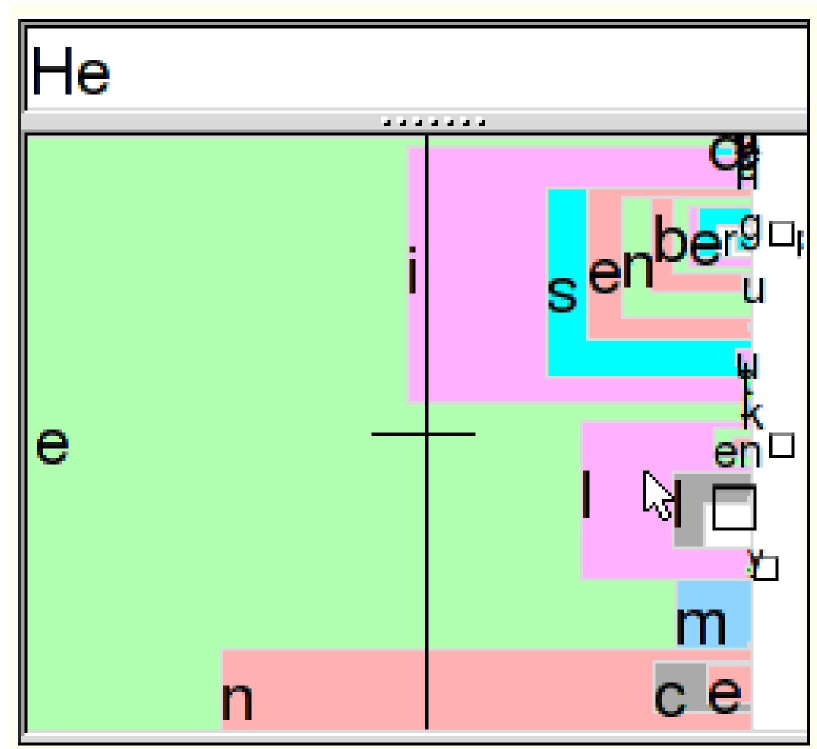
Canesta projection keyboard

http://www.youtube.com/watch?v=uDf_Go7dBNs

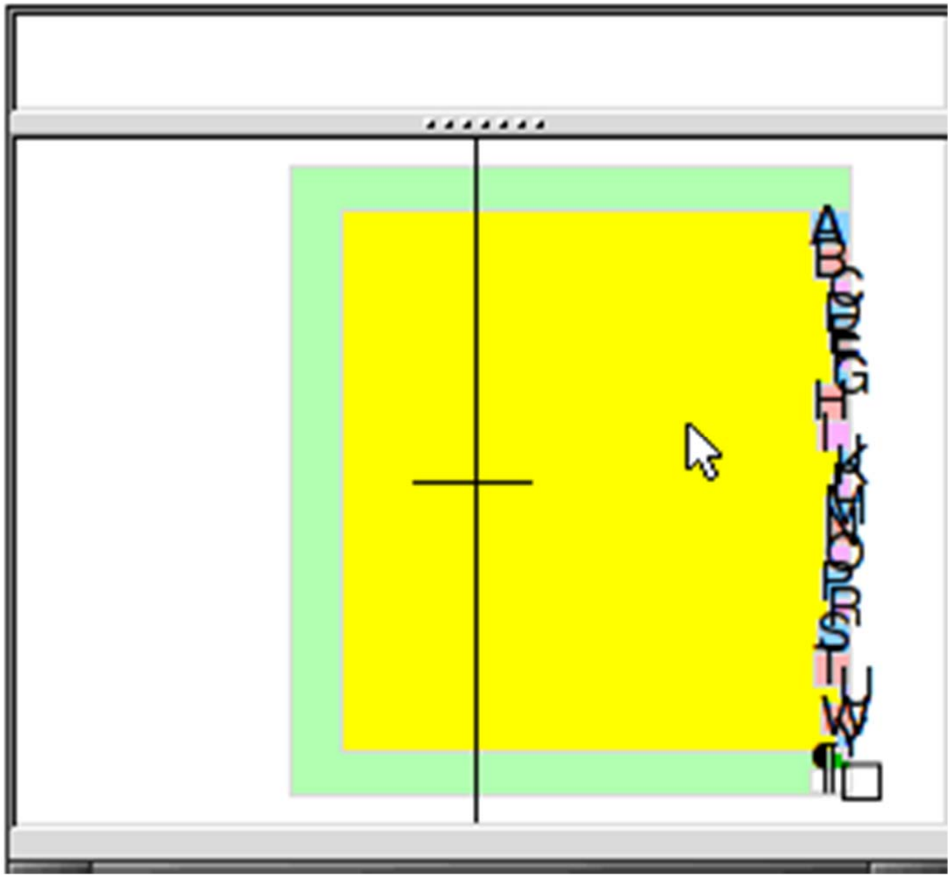


Dasher (Ward et al. 2000)

- Instead of moving to create text, let text move to you.
- Letter regions expand outward toward cursor.
- Letter region sizes are based on language frequencies.
- Speed of expansion is governed by cursor x-coordinate.



Dasher video



http://www.inference.phy.cam.ac.uk/dasher/images/largedasher.gif

Handwriting recognition

- Use advanced pattern recognition algorithms to support natural handwriting input.

(MacKenzie & Tanaka-Ishii 2007)

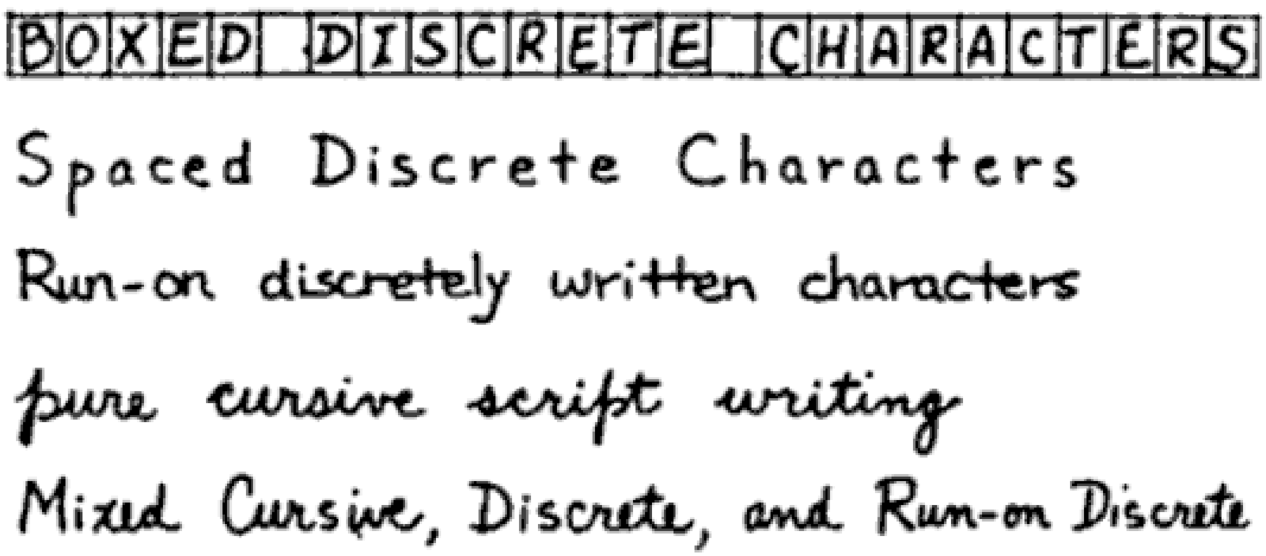


FIGURE 6.1 English writing styles for computer input (Tappert *et al.*, 1990).

Segmentation problem

- How to tell where one letter ends and the next begins?

clearly beloved

Ambiguity

- Context is essential for recognition.

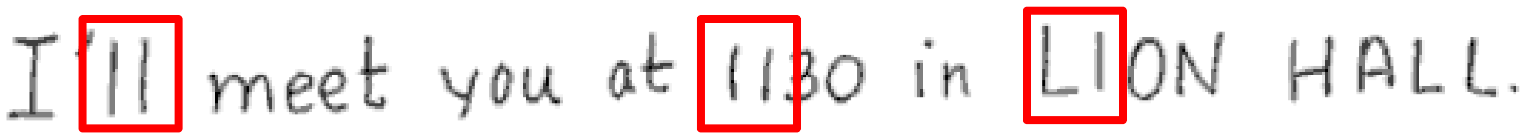
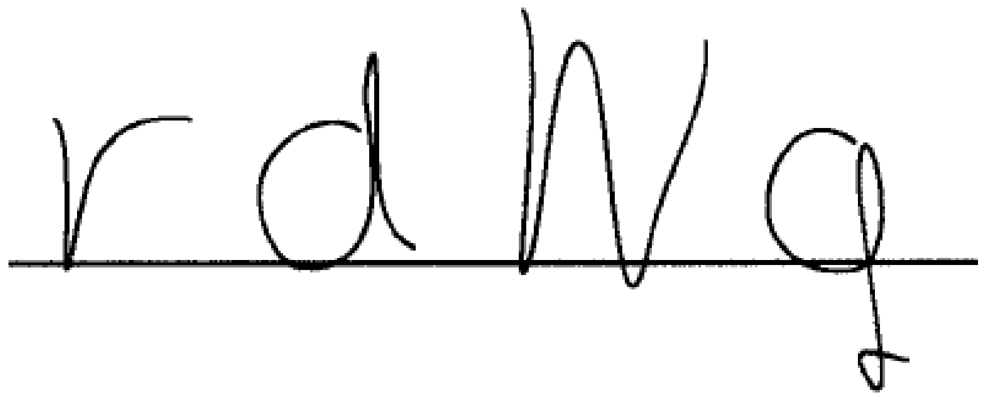


FIGURE 6.2 Different characters with the same shape.

Ambiguity

For example, in the following figure, is the first letter an 'r'



or a 'v'? The second an 'a' or a 'd'? The third an 'N' or a 'W'? The fourth a 'g' or a 'q'?

(Goldberg & Richardson 1993)

Apple Newton

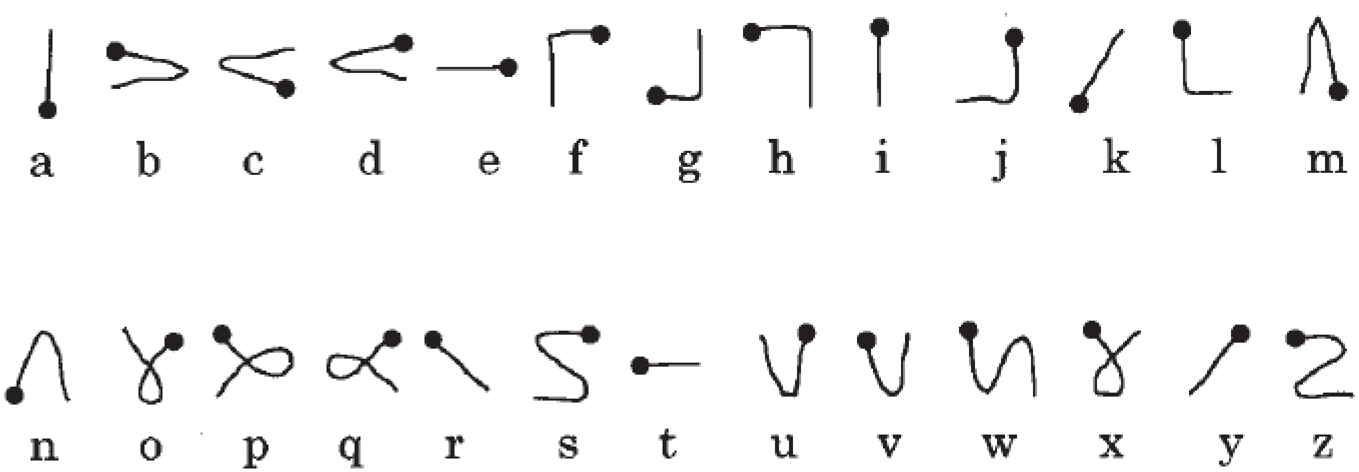
- Boasted the first commercially available handwriting recognition.

(The Simpsons)

https://www.youtube.com/watch?v=u6qxixgQJ4M&t=10s

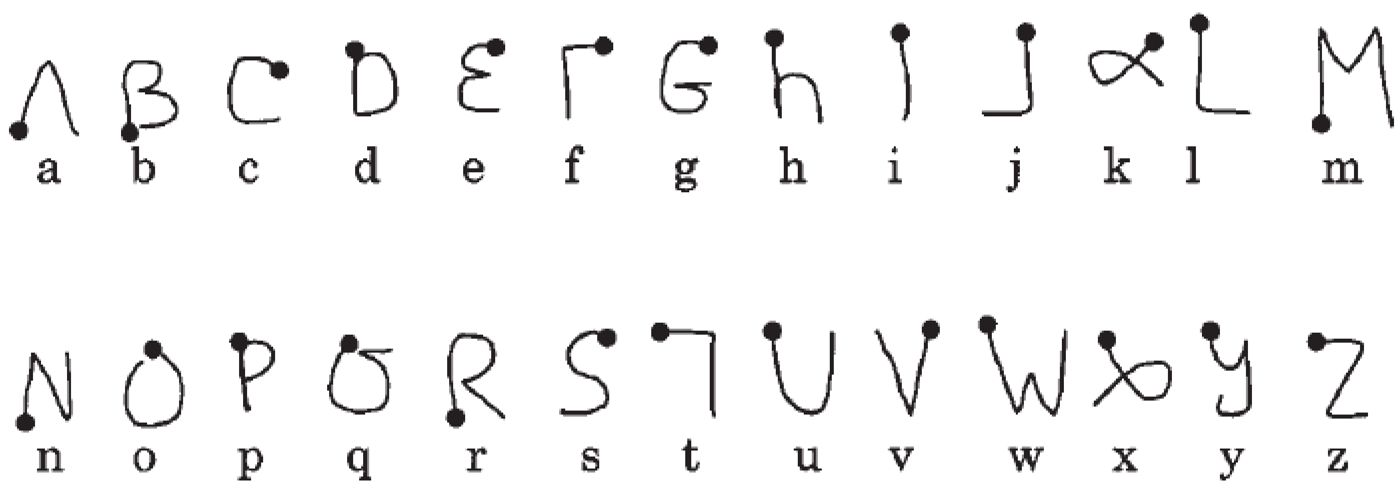
Unistrokes (Goldberg & Richardson 1993)

- Solved the segmentation problem by having each letter be a single stroke.
  - "Lift" is the segmentation signal.

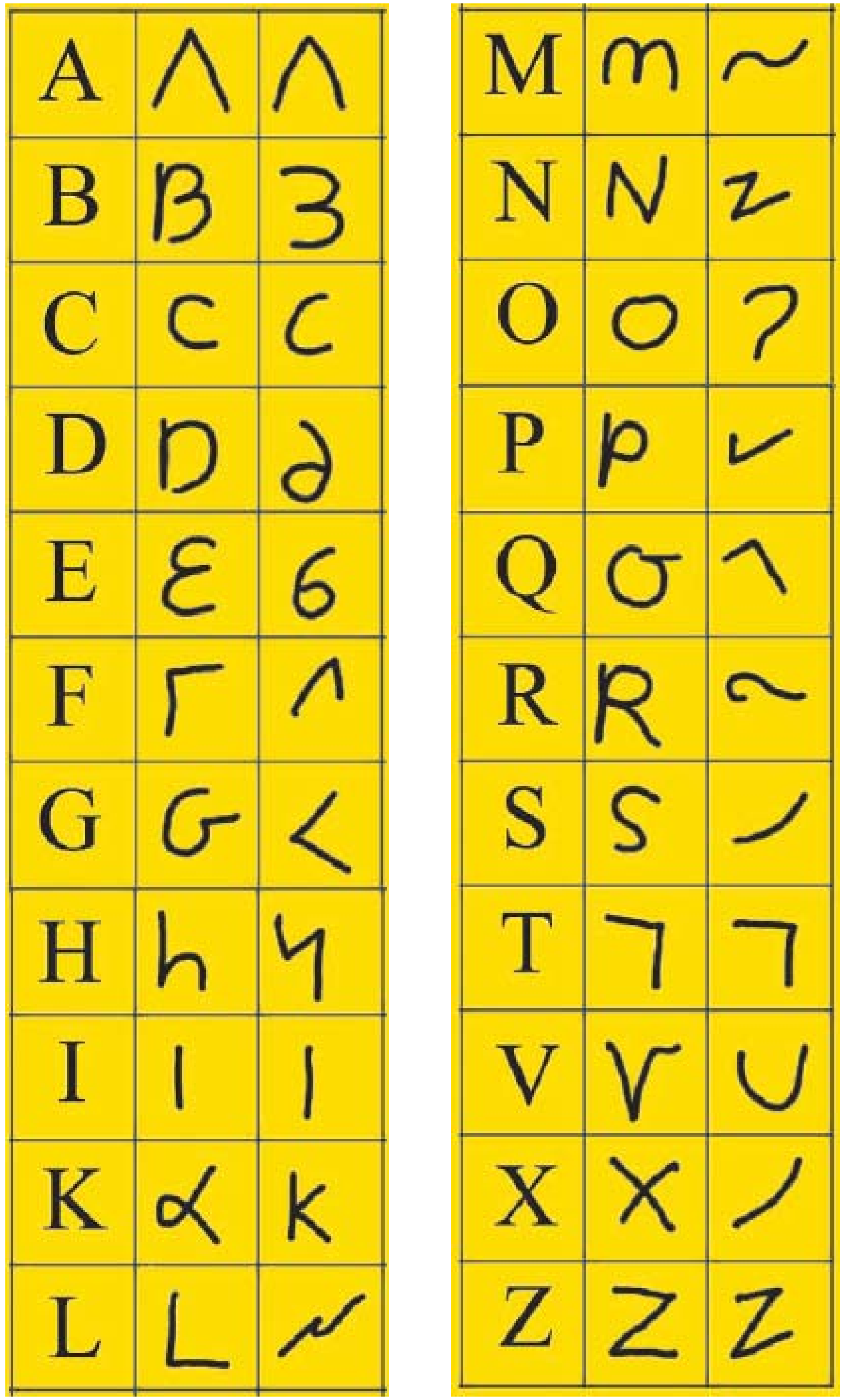


Graffiti (Palm, Inc. 1995)

- Unistroke alphabet much more learnable and letterlike than Unistrokes. ~25 wpm



Beside each modern alphabetic character appear the *Graffiti* and *Notae Tironianae,* symbols that represent it (middle and right columns, respectively). Notae Tironianae, likely the first single-stroke short-hand, was developed in 63 BC by a freed slave of Cicero. Illustration by the author based on: Panati, C. (1984). ***Panati's Browser's Book of Beginnings***. Boston: Houghton Mifflin Co. p.81



Buxton, W. (2005) "Piloting through the maze." *interactions 12 (6), November/December 2005, p. 10.*

SHARK^2 (Kristensson & Zhai 2004, commercial name: *ShapeWriter*)

- Stylus keyboards are easy to learn.
- Letter-like unistrokes are too slow to perform.
- Combine keyboards with *word-level* unistrokes.
- Strokes are defined by their pattern on the underlying keyboard.

http://www.youtube.com/watch?v=WtlyuuYmFN0

Swype

=@#

htc
Sprint
Swype is quite quick
quick
wick
erick
stock
sick
SYM

Evaluation issues

- Speed: words per minute (WPM)
  - Technical definition
- Accuracy: how do we define this?
  - Uncorrected, corrected, and total errors
- Should subjects compose or transcribe text?
- How should they be trained, if at all?

Text composition *vs.* transcription

- Text composition
  - Typing in on an initially blank screen
  - What are the challenges regarding measuring text entry performance?
- Text transcription
  - Copying a presented phrase
  - Does this solve all the challenges? Present new ones?

Transcription schemes

- Disallow errors
  - No mistakes appear
  - Beeps played
  - Error "chunks" common
  - Backspace is irrelevant
- Allow errors, prevent error correction
  - Errors appear, and writer must re-synchronize with presented text

```
P: See spot run
T: See x
```

Not allowed, does not appear, cursor stays put

```
P: See spot run
T: See pot rrun
```

Omitted "s" causes other letters to be off, user re-synchronizes at "r".

Transcription schemes cont.

- Force users to correct all errors before a phrase is treated as finished
  - This can dramatically reduce WPM
- Redo all trials with any errors
  - Insist on perfect transcription
- Simply ignore errors

```
P: See spot run
T: Sea spot run
```

User has to go back and
fix this before the trial ends.

```
P: See spot run
T: See spot run
```

```
P: See spot run
T: Sed ahsi rqh
```

Unconstrained text entry

- New paradigm that allows for natural transcription without artificial constraints
  - However, only backspace may be used
  - Relies on comparison algorithms for determining errors

```
P: See spot run
T: See rspot run
```

Will correctly identify the “r” as an error but not what follows it, even though it is out-of-sync.

P, *T*, and *IS*

- *P*: presented string for transcription
- *T*: transcribed string the user enters
- *IS*: input stream, the record of all keystrokes the user makes in creating *T*

How many errors?

```
P: the quick brown
T: the quicxk brown
              ^^^^^^^^^

IS: f<tn<he p<qul<ik<cxk bfo<<rown
```

How many errors?

```
P: quickly
T: qucehkly
```

Optimal alignments

```
P: quickly
T: qucehkly
```

All have 3 errors.

```
P1: qu-ickly
T1: qucehkly

P2: qui-ckly
T2: qucehkly

P3: quic-kly
T3: qucehkly

P4: quic--kly
T4: qu-cehkly
```

Fundamental error types

- Substitution
- Insertion
- Omission (aka, deletion)

P_1: qu-ickly
T_1: qucehkly

P_2: qui-ckly
T_2: qucehkly

P_3: quic-kly
T_3: qucehkly

P_4: quic--kly
T_4: qu-cehkly

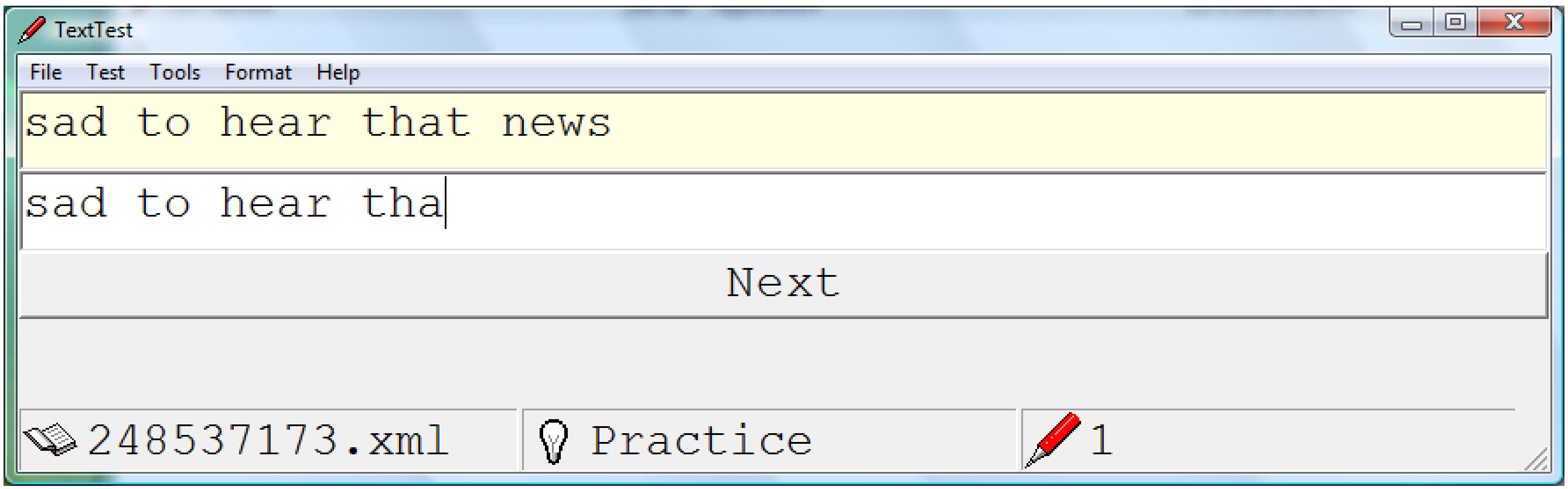
Minimum string distance (MSD)

```
MSD-MATRIX(P, T)
D ← new matrix of dimensions |P| + 1, |T| + 1
for i ← 0 to |P| do
    D[i, 0] ← i
for j ← 0 to |T| do
    D[0, j] ← j
for i ← 1 to |P| do
    for j ← 1 to |T| do
        D[i, j] ← MIN(D[i – 1, j] + 1,
                      D[i, j – 1] + 1,
                      D[i – 1, j – 1] + P[i – 1] ≠ T[j – 1])
return D[|P|, |T|] and D
```

Compute optimal alignments

```
ALIGN(P, T, D, x, y, P', T', ref alignments)
if x = 0 and y = 0 then
    alignments ←+ (P', T')  // add a new aligned pair
    return
if x > 0 and y > 0 then
    if D[x, y] = D[x – 1, y – 1] and P[x – 1] = T[y – 1] then
        ALIGN(P, T, D, x – 1, y – 1, P[x – 1] + P', T[y – 1] + T')
    if D[x, y] = D[x – 1, y – 1] + 1 then
        ALIGN(P, T, D, x – 1, y – 1, P[x – 1] + P', T[y – 1] + T')
if x > 0 and D[x, y] = D[x – 1, y] + 1 then
    ALIGN(P, T, D, x – 1, y, P[x – 1] + P', "-" + T')
if y > 0 and D[x, y] = D[x, y – 1] + 1 then
    ALIGN(P, T, D, x, y – 1, "-" + P', T[y – 1] + T')
```

TextTest (Wobbrock & Myers 2006)



Want to know more?

- Wobbrock, J.O. and Myers, B.A. (2006). Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction 13* (4), pp. 458-489.
- http://depts.washington.edu/ewrite/