WORKING NOTE AC105

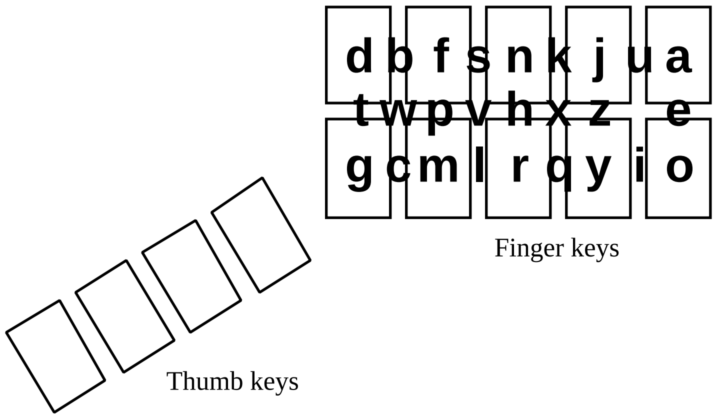
Alan Creak
24 October 1996

ANALYSIS OF A ONE-HANDED CHORD KEYBOARD.

I inspect the specifications for a proposed one-handed keyboard, and decide that it could be done better. I also give my notation a further work-out.

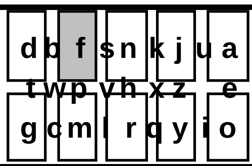
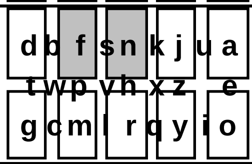
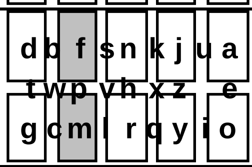
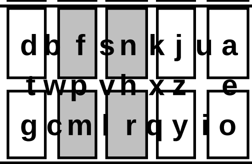
THE DEVICE.

Bequaert and Rochester were granted a patent[3] for a one-handed keyboard with a number of novel and interesting features. The keyboard has ten finger keys and four thumb keys, and looks – more or less – like this :

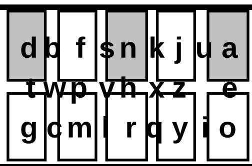
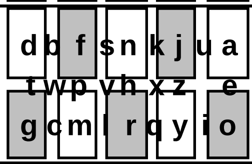
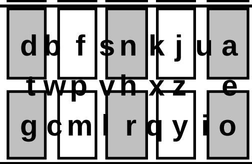
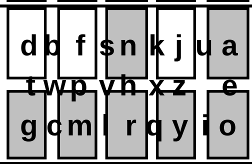


For the most part, I shall discuss the operation of the finger keys, but the same principles, with obvious and minor modification, apply to the thumb keys.

The finger keys can be used as a chord keyboard in two distinct ways. First, a combination of keys which can be pressed with a single finger identifies a single character; that is why there are characters between the keys in the diagram. This is how it works :

	means	*f*
	means	*s*
	means	*p*
	means	*v*

where the grey keys identify those pressed in each case. In the second type of chord, key combinations which cannot be pressed with a single finger can be used; these are interpreted as character sequences :

	means	*dna*
	means	*gfrjo*
	means	*the*
	means	*chia*

The first row illustrates the general principle, with several characters selected by key depressions and read from left to right in sequence. The second row shows one of the longest strings which can be produced; this is a simple example, in which all five characters are separately selected and no ambiguity need be resolved. The third case illustrates the use of multiple-key characters, but once again the interpretation is simple.

The final row is harder to interpret, because there are several possible combinations of characters which give the same pattern of depressed keys : *gmhye*, *glnia*, *chye*, and others are possible interpretations, and if overlapped combinations are considered there are even more. An unambiguous meaning is associated with such a pattern by following a decoding algorithm. This is embodied in the rather complicated electronic logic described in the patent document, but in essence works columnwise from left to right, taking in as many pressed keys at each step as possible. In the example

- *g* is pressed in the first column, but that can be combined with the next pressed key (*m*) to give *c*;
- in the third column, both keys are pressed, so *h* results; this combination could be extended to the next column if both the keys there were pressed (to give *x*), but they are not, so *h* remains.
- in the fourth column, *y* is pressed, and can combine with *o* to give *i*;
- finally, *a* remains unused in the fifth column.

The same left-to-right procedure is used to determine the meaning of any key combination.

The thumb can be used to modify the character string derived from the finger keys in several ways, such as adding spaces before and after, selecting different characters sets, and changing case. More interesting, it can select a mode in which the sequence of characters from the keyboard is reversed – so with this mode selected, the sequences from the example would be *and*, *ojrfg*, *eht*, and *aihc*. The appearance of *and* is interesting, and obviously useful (and unlikely to be accidental); both *and* and *the* can be entered with a single action. The second and third examples have no special features; but the fourth is more interesting in that, though the characters are ordered from right to left, the scan which determines the set of characters still operates from left to right. (A reversed scan would give the sequence *eqnc*.)

COMMENTS.

The most obvious comment is that text is not too easy to encode; perhaps with practice the maximum speed could be achieved, but learning does not appear to be simple. Looking forward to determine what letter sequences can be encoded would be hard. Trying to work out the details of complicated reverse combinations might well be very hard, because of the interaction between a forward scan to determine the interference and a backward scan to work out the possible combinations. It is therefore rather unlikely that high speeds could be achieved by a beginner. The same is true of all keyboards; until the translation of intention into finger movements becomes automatic, maximum speed cannot be attained. With this keyboard, though, the possibility of using chords introduces an additional element – the possible letter combinations – to be learnt, and these are unlikely to be useful unless their use also becomes automatic.

It seems likely that with time a practitioner could learn to use common patterns. Whether or not that would be useful depends on how easy it is to discern the patterns in the text in time to use the abbreviations. Short of learning all words independently, some cognitive analysis must proceed in parallel with the typing. Do conventional typists learn to recognise – say – common prefixes and suffixes ? – and, if they do, are they able to make use of the associated muscular patterns if the same group of characters appears in the middle of a word ? (Consider *ca**tion**ic*, *ped**anti**c*, etc..) The use of Braille contractions is another analogy, and, as with this keyboard, there are rules which must be satisfied over and above recognising the letter patterns, and which must be satisfied between recognising the text and typing it. As an example, consider typing the word *through*. Common sequences within the word which might well be learnt by practice are *th*, *ro*, and *gh*; but *th* and *ro* cannot be keyed at the same time, because *h* and *r* interfere. On the other hand, *tho* (in *though*) can be entered in one stroke.

My example of accidental prefixes and suffixes is interesting, because neither of the combinations I used in the illustration (*tion*, *anti*) is accessible with the one-handed keyboard. Generally, because of the interaction between the left-to-right (or right-to-left) interpretation and the keyboard layout some patterns must inevitably be inaccessible. All in all, the inventors seem to have done quite a good job at placing characters, perhaps by considering digram frequencies. One might query the placing the vowels at the right-hand end of the keyboard, because this immediately rules out many common letter sequences with internal vowels, but perhaps there is a good reason. The arrangement does lead to a possible encoding rule : learn to react quickly to any consonant-vowel sequence. That alone would significantly reduce the number of keystrokes necessary to type ordinary text, and with the addition of a few common consonant combinations might prove useful.

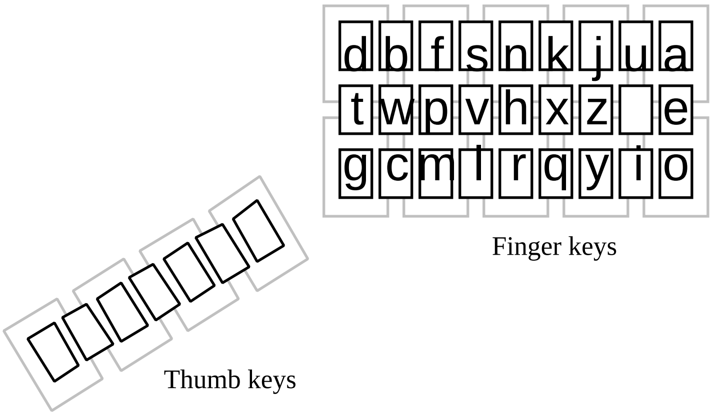
The ordering constraint is perhaps inevitable, but this keyboard introduces another complication. Many potentially useful combinations of characters represented by multiple key depressions are ruled out by the interference between their key requirements. Some examples – by no means an exhaustive list – of this sort of interference are *cl*, *ph*, *sh*, *sl*, *sn*, *sk*, *oi*, *ea* (in fact, all vowel combinations except *uo*, *ia*), and many plural and present tense endings (*ds*, *bs*, *ws*, *fs*, *ps*, *vs*, *ls*, *hs*, *rs*, *ks*, *xs*, *js*). It is possible that some of these difficulties could be resolved by redesigning the layout of the keyboard, but in any case some such interference is inevitable because of the use of multiple keys to represent single characters.

Despite these criticisms, the possibility of using chords of more than one letter necessarily means that the attainable typing speed is greater than it would be for a comparable keyboard in which such chords were not allowed. It's therefore necessarily no slower than one key at a time, and every advance on that mode of operation is gain. It is less than clear that the maximum speeds estimated in the patent document are attainable without extensive training, and to achieve the absolute minimum number of keystrokes at the maximum possible typing speed would require considerable cognitive effort throughout the activity, which might be prohibitive.

But we can do better.

AN ALTERNATIVE.

Once the novelty has worn off and you start to look at the keyboard more critically, you observe a very curious fact. In terms of the Creak and Sheehan view of communication[4], the finger keyboard described is a digitiser for spatial position; it divides an area of space into ten subareas, each of which can be associated with some meaning by the surrounding system. The device of simultaneously depressing adjacent keys is a means of obtaining a finer segmentation of space, in this case into 27 subareas instead of ten. I thought of that too[5], but I was trying to get more out of the standard keyboard. Bequaert and Rochester are inventing a keyboard from scratch – so why don't they just build a 27-key keyboard ? It would look like this :



With this keyboard, we can get any character combination, *without any interference* – and we can introduce a new rule (top-to-bottom, then left-to-right) so that an unambiguous and simple order can always be used to determine the sequence of multiple-character chords. (I'd want to change it to top-to-bottom, right-to-left for reversed sequences.)

The redesign makes essentially *no difference in physical operation*; the same finger positions are used in exactly the same ways. The distinct keys might make it rather easier to find the required finger positions. (It's interesting – not to say bizarre – that the patent suggests that depressions might be moulded in the ten-key keypad the better to identify each finger position. Why didn't they think of the 27-key solution ? – or had that been patented already ?) The only possible change that I can imagine is that there might be slight differences in the relative areas of effective keys, which are all the same with the 27 discrete keys but dependent on finger shape and pose for the ten-key construction.

So far as encoding text is concerned, the main difference is that the interference problems have vanished. As each letter now has its own key, there is no conflict of interpretation of any pattern of key depressions. The constraints imposed by order of interpretation remain, and are probably inevitable; there must be some ordering rule to determine the meaning of a chord. But I think that even with the new top-to-bottom convention no chord possible with the original version becomes impossible with the 27-key design. (As the basic left-to-right and right-to-left scans are unchanged, the only possible source of problems is the top-to-bottom scan; but any such scan on the new keyboard is equivalent to combining corresponding key actions on the top and bottom rows of the original keyboard, which would be interpreted as some form of chord meaning a different character.) This must reduce the cognitive complexity of encoding, and certainly makes available many of the character sequences which were ruled out by the interference; the result might be to make it more practicable to learn to encode text directly to the keyboard rather than simply learn a large set of standard hand poses, but that's a possibility rather than a certainty.

On the mechanical and electronic side, there are more mechanical components, but far less electronics. Mechanically, more keys are necessary, though there will be less wear per contact. In any case, the result is nothing like as big as a typewriter keyboard, and they're pretty reliable. Electronically, there is no longer any need to decode the potentially interfering chords for single letters; the patent document spends a lot of space on describing just how that should be done. What's left is simple buffering, detection of whatever signal is used to identify the completion of the chord, and transmission of the character sequence.

INFORMAL ANALYSIS OF PERFORMANCE.

It is interesting (though not as interesting as I expected when I started) to analyse the properties and behaviour of the keyboard more or less according to my method[1], as refined in the light of some experience[2].

This is only a partial description. The items I list in the leftmost column are chosen more or less arbitrarily, but it's sensible to include enough steps to give (what seems to be) a complete description of the operation at some level. I assume that the sender is conscious of an attention point on the keyboard which starts at the left-hand end of the keyboard at the beginning of the session and after any chord has been transmitted. I've ignored backward scans and other thumb functions entirely.

Item name	*Locus*	*Vocabulary*	*Description*	*Remark*	
				10 keys	*27 keys*
S-notion	S	Thoughts	What S (the sender) wants to say.		
→ sentence	S	Any word(s)	Chosen by S.		
→ letter sequence	S	Letters	How to spell the next word.		
→ next letter	S	Keyboard actions	The next letter to type.		
→ search down and to the right	S – K	(action)	Search the key labels on the keyboard.	Vocabulary is "really" muscular and cognitive actions to scan the table.	

If the letter is found –

→ check for interference	S – K	(action)	Check the keying rules.	Probably largely cognitive ?	*DOESN'T APPLY*

If there is no interference –

→ move finger to position	S – K	One of 27 positions	Press the key(s).	Largely muscular; don't move the other fingers.	
→ repeat letter selection	S	Possible actions	S chooses what to do next.	A control action.	

If there is interference, or the letter is not found –

→ raise fingers	S – K	(action)	Mark chord complete.	Transmit the chord.	
→ repeat letter selection	S	Possible actions	S chooses what to do next.	A control action.	
→ transmit keyboard signal	K – H	N letters encoded together	The chord is sent to the analysis Hardware (H).	10 channels; at least N keys are pressed.	27 channels; exactly N keys are pressed.
→ resolve ambiguities	H	Electronic logic	Sort out the letter combination.	A standard hardware sequence.	*DOESN'T APPLY.*
→ identify the letters	H	Electronic logic	Work out the intended letter sequence.	A standard hardware sequence.	
→ send the letter sequence	H	Standard electronic code	Transmit the sequence as from an ordinary keyboard.		

The two sequences are almost identical, with only two differences between the descriptions. One is cognitive : the operator need not check that different letter representations overlap. The other is physical : there are different numbers of channels in the two cases, and there is therefore no need for machinery to resolve the multiple-key codes. It seems quite likely that a minor additional complication to the keyboard could result in a product which is significantly easier to use.

BRIEF NOTE ON THE ANALYSIS.

This example shows that my method falls short in its provision for branching and parallelism. Though parallel operation has certainly been considered, because all levels of activity are assumed to occur at the same time, the topics haven't turned up in this form before; is that odd ?

I think that the reason might be that in this example I'm concentrating much more on the sequence of operations, rather than solely on the possibility of managing the communication. In trying to compare the two keyboard designs, the salient feature is not that one design can manage the communication while the other can't – clearly, both are usable in principle – but that one will be more efficient than the other. This leads me into questions of speed, so I become interested not only in the possibility of encoding but also in how it is done – so I take into account details of how the chord is made and how it is ended. At this level, the sequence of operations is more important, and the method really doesn't cope too well.

Should it cope ? On the whole, I don't think so. Its particular virtue is in checking that the information of a message can be preserved and transmitted at each stage of the operation, and in doing so in an informal manner which allows one to explore a variety of issues which might arise. More formal methods should be used to explore efficiency, because some analysis of specific methods is necessarily entailed. This is a possible outcome of a so-far-hypothetical formalisation of this technique[6], but that might never happen.

REFERENCES.

1 : G.A. Creak : *Reaching beyond words in rehabilitation computer systems*, unpublished Working Note AC96 (September, 1996).

2 : G.A. Creak : *Analysing WordKeys*, unpublished Working Note AC103 (September, 1996).

3 : F.C. Bequaert, N. Rochester : *One-handed keyboard and its control means*, US Patent 4042777 (August, 1977).

4 : G.A. Creak, R. Sheehan : *The representation of information in rehabilitation computing*, Auckland Computer Science Report #54, Auckland University Computer Science Department, July 1991.

5 : G.A. Creak : *Multiple keying for faster communications*, unpublished Working Note AC91 (September, 1994).

6 : G.A. Creak : *Interactors in rehabilitation system design*, unpublished Working Note AC97 (August, 1996).