WORKING NOTE AC110

Alan Creak
12 December 1996

NOVEL APPROACHES TO USING KEYBOARDS

Possibilities for using computer keyboards as input devices for people with disabilities are considered. By relaxing the conventional rules for keyboard use, other potentially useful modes can be found. I discuss two such modes in some detail, and comment on some experiences with a prototype implementation of one of them. The method is judged promising, and further investigation is planned.

The conventional typewriter keyboard is very widely used, and it is difficult to buy a microcomputer without one; in effect, if you have a computer then the keyboard costs nothing. That alone is a recommendation for using a keyboard as an interface device for rehabilitation computing systems if possible, for free interface hardware is welcome to anyone. This is particularly so for those with limited income, as is often the condition of people with disabilities.

In practice, cheapness is not the only consideration; however cheap the keyboard, it's worthless if people can't use it, and much effort and ingenuity has been expended in providing alternative interface devices for people who find it physically very difficult to use a conventional keyboard. Others can use a keyboard, but with great difficulty, and perhaps therefore only very slowly.

But why should it be so hard to use the keyboard ? For those with very limited control of their hands, or with other constraints which make effective use of the keyboard physically impossible, the answer is obvious, and many people who don't use keyboards therefore communicate instead through various sorts of switch device. But if you can use a switch, why can't you use a keyboard ? What is a keyboard, after all, but a set of switches ?

WHAT DOES A KEYBOARD DO ?

A keyboard certainly is a set of switches, but it is also a position sensor for two-dimensional space. When you use a keyboard to type a single character, it is the position of your finger in space which determines which character you select. The geography of a computer keyboard divides an area of space into 80 or so regions, one for each key (say, *K* regions), each of which has a distinct meaning identified in the first instance by the label on the corresponding key, and thereby quantises your finger position in the sense that moving it a small distance in one direction or the other doesn't change the meaning. In terms of a convenient set of attributes[1], we can describe the input signal to the keyboard in this way :

Medium :	Two-dimensional space.
Signal :	Quantised, *K* defined areas.
Density :	*K*-vectors of binary digits; exact; arbitrary speed.
Quality :	Excellent.

Other similar tables will appear later, so it is worth explaining this example in some detail. Any signal must be embodied as a set of values of the disposable variables of some communications *medium*, which in this case is two-dimensional space. There are commonly many ways in which a medium could be used to form *signals*; one way with this medium is simply to define two coordinate axes and use the *x* and *y* coordinates so defined, but for a keyboard it is more useful to think in terms of dividing the space into a small region for each key (and a few more), each of which corresponds to one component of the total signal. The *density* of the signal gives some information on how the signal is used, and particularly on the amount of information which the channel can carry – this is the product of the amount of information carried by one instance of the signal, and the rate at which signals are conveyed. In this case, each key might be up or down, and all the keys move independently, so the complete signal includes one binary digit for each key. Finally, so far as the keyboard is concerned, the signal is always of excellent quality – each key is unambiguously either up or down. (Whether the hand or other manipulator which is pressing the keys is issuing information of equally excellent quantity is another matter, but here we are discussing the keyboard itself.)

Notice that there is nothing in this geometrical view of the keyboard to limit the number of keys pressed at once to one or two. That reflects the structure of the keyboard itself; there's nothing to stop you

using all your fingers and thumbs at once, or the flat of your hand, or whatever combination of keys you can press with any implement you choose. The limitations which are in practice imposed by most – perhaps all ? – computer keyboards are consequences of what happens later in the processing stream; I shall return to this topic later.

HOW IS IT USUALLY USED ?

In practice, that isn't the way we usually use keyboards. Instead, we press only one or two keys at a time. In part, that's a habit which has stayed with us since the days of mechanical typewriters, when pressing two or more printing keys at once had unhappy consequences (shift keys worked in a different way), and, anyway, why would you want to press lots of keys at once ? – for all the symbols you might need are there on the keyboard, and the more complicated combinations don't mean anything useful.

To see just how hands interact with keyboards, it's illuminating to analyse the actions of the hand in the same way as the function of the keyboard is described in the table above. For our purposes, hands, like keyboards, are communication channels. They can be very effective communication channels, as in sign languages, where the hands convey most of (though not all) the information. The transfer of information from hand to keyboard is the focus of this discussion, and if the transfer is to be effective there must be some degree of matching between the channels involved. Generally, we can describe hands using the same terms as we did for the keyboard.

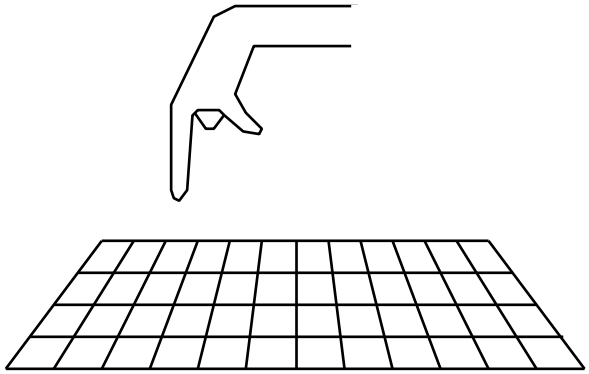
Medium :	*N*-dimensional space.
Signal :	Continuous, one channel for each of the *N* joints.
Density :	*N*-vectors of real numbers; pretty accurate; variable speed.
Quality :	Variable.

That definition is constructed by thinking of a hand conformation as the result of a large number (*N*) of independent individual muscular movements. It is true so far as it goes, but – like a description of a keyboard in terms of *x* and *y* coordinates – doesn't capture the way we really use the communication channel. Instead, just as with the keyboard, we don't worry about the exact values of the real coordinates, but identify bigger areas of the space which are comparatively easy for us to identify. Once again, we quantise the possibilities, separating out certain recognisable features, so a more practical description might look like this :

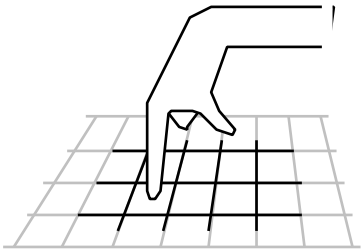
Medium :	Position	Orientation	Pose
Signal :	Quantised; one channel.		
Density :	Recognised set; variable speed.		
Quality :	Often good – but can be compromised.		

Three distinct, and more or less independent, channels are identified : the position of the hand in space, determined by the shoulder and elbow muscles; its orientation, determined by the wrist and forearm twisting muscles; and its pose, the shape of the hand itself, determined by the rest. Each of these channels is then used in much the same way. A set of significant values for each is identified by convention, and any actual hand conformation classified as an example of some combination of these standard values for each of the channels. This makes signing far easier, because provided that we put our hands into something reasonably close to the ideal position, the sign will be classified correctly, and the right meaning will be conveyed.

What does this view show us when we consider how hands are conventionally used with keyboards ? As a simple example, take the case of a one-finger typist typing only lower-case characters. The primary restrictions on hand configurations are summarised in this diagram :



Of the many possible poses and orientations, only one of each is significant – one finger must be pointing downwards. The position channel remains essentially free, which is what one might expect to match the freedom of the keyboard's position channel, but the freedom is qualified by the requirement that the finger must meet the keyboard on one of the keys, not on the boundaries between keys :



That description was presented in terms of a one-finger typist in the interests of simplicity, but analogous conclusions follow from a corresponding analysis of touch-typing. The details are different – two hands are concerned, their positions are almost static, and several poses are important, but only poses with one finger tip pointing down are significant, and the orientation is once again essentially fixed throughout.

I suggest that the foregoing discussion provides support for two conjectures on the interaction between hands and keyboards.

- Despite their widespread use, keyboards are not well matched to hands. Only one of the three communication channels of the hand is used to any extent when operating a keyboard. It might be a little fanciful to compare the rate of transfer of information attained by average typists and average sign language users, but the additional channel capacity of the sign language must contribute something to the fluency and speed of communication. Perhaps typing shows up so well in comparison only because it profits from the extraordinary agility attainable by fingers.

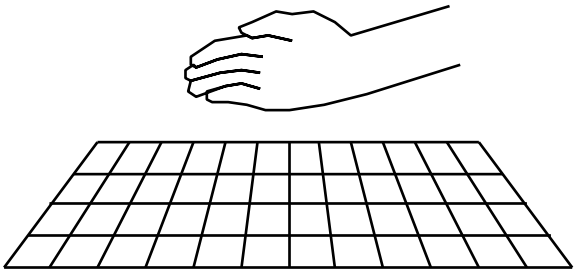
- Many people who say that they cannot use a keyboard really mean that they cannot conform to the arbitrary rules conventionally imposed. Perhaps if the rules were changed, more people would be able to use keyboards effectively as means of communication.

CHANGING THE RULES.

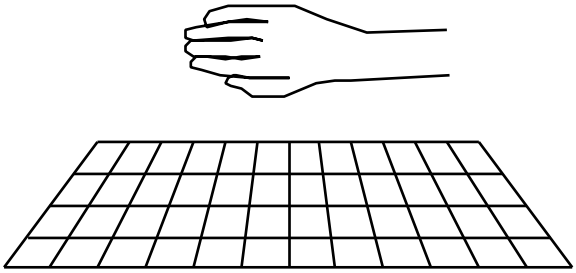
It becomes clear that the keyboard, as usually used, exploits only a tiny proportion of the capability of the hand as a communication channel. Some mismatch is inevitable, as the keyboard can only detect an object which touches it, so the hand position must be restricted to the region in which it touches the keyboard. Further, as the keyboard is essentially two-dimensional, it can never detect the whole three-dimensional pose of a hand. Even so, there is certainly potential for alternative approaches in which a keyboard is used to detect a wider range of hand conformations.

To see why such approaches might be of interest, consider again the second description of the hand as a communication channel, where the quality of each component channel is described as "can be compromised". While most people's hands are agile and flexible, many people with various sorts of motor disability, arthritis, or other condition cannot form their hands into arbitrary poses; in extreme cases, only one pose might be accessible without serious discomfort. If this pose does not coincide significantly with the standard pose demanded by the usual keyboard rules, then typing can become a demanding or painful

or even impossible activity. A keyboard which could accept signals produced in other poses could be very useful to someone in such circumstances. For example, it might be possible to approach a keyboard with a pose like this :

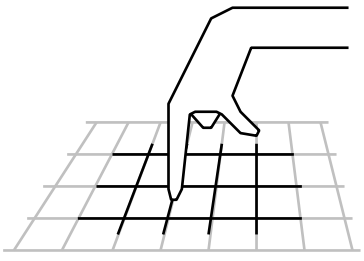


By an analogous argument, it might be possible to exploit the orientation channel of the hand. Using the same pose as in the previous diagram, but turning the hand over, this approach might be possible :



Provided that the signals generated by the keyboard with the hands in the different orientations are distinguishable, exploiting these channels could give comparatively easy access to a computer system in two distinct modes, which might be used for upper case and lower case characters.

There still remains an element of unexploited freedom in the position channel. As mentioned above, it is assumed in a conventional keyboard system that the finger will touch a single key near its centre. If this condition is relaxed, we can consider the possibility of using effectively simultaneous depressions of adjacent keys as acceptable signals, thereby significantly increasing the number of detectable regions in space :

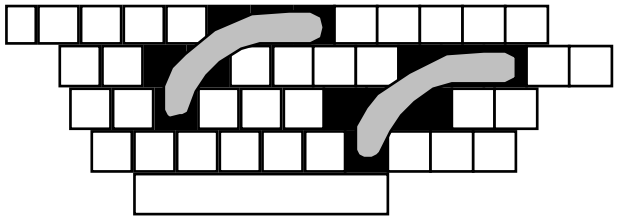


Such a method might be useful to someone who could place a finger, or other manipulator, in an intended position quite precisely, though not very quickly; many people who use head pointers might be able to use this type of encoding. (Good precision is necessary, and certainly attainable in some cases; consider the achievements of many mouth-painters.)

TWO POSSIBILITIES.

In more practical terms, these considerations lead to two specific proposals for interfaces that might be found useful as communication aids.

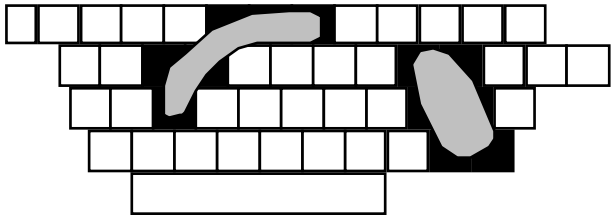
The first proposal is a development of the notion of relaxing the standard requirement for a rigid pose of the hand when using the keyboard. If any pattern of keys can be detected, then it will be possible to distinguish between the patterns made by adopting a single pose, and applying it at different places on the keyboard. For example, the patterns below are produced by applying a hand in the pose with thumb uppermost shown in the earlier diagram to a particular keyboard :



(This diagram, like the next, shows the real result of placing my hand on the conventional part of the keyboard of my computer.)

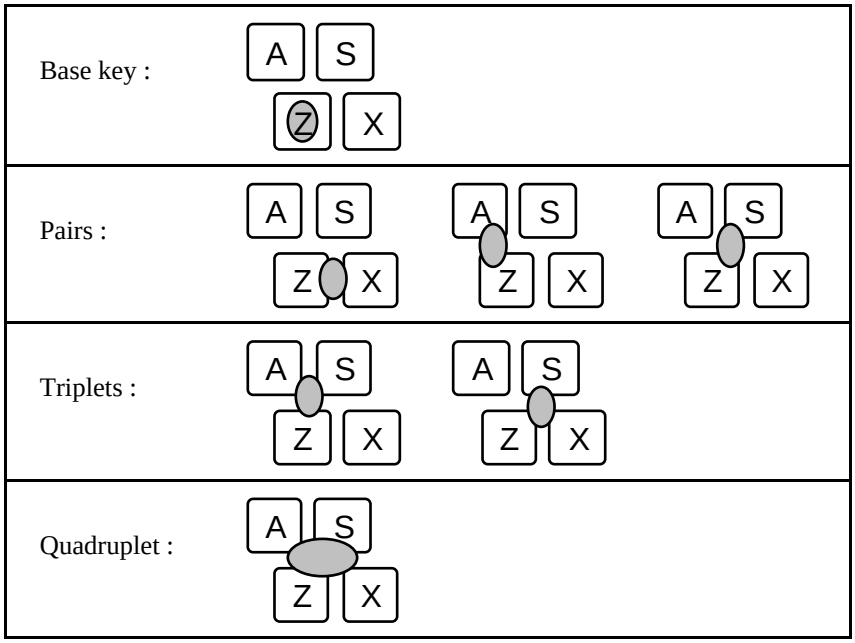
By adopting a convention to decide which key is intended to be identified by each pose – the obvious convention is to select the key pressed by the end of the little finger, which is the lowest key shown as depressed on the diagram – this gives access to all keys of the keyboard without demanding that the typist should adopt some inconvenient pose which might be fatiguing to maintain and significantly slow down the typing speed.

If it is possible to use the hand's orientation channel as well, the repertoire can be expanded. For example, extending the principle to accept patterns produced from a second orientation, another set of signals can be given. This is a pattern produced using the thumb-down orientation illustrated earlier, with a thumb-up example for comparison :



There are some practical complications, but – with the exception of the rollover problem, which I shall discuss below – these are not obviously serious. An example is the likelihood that not exactly the same pattern will be produced every time the hand is lowered onto the keyboard in what is intended to be the same pose, position, and orientation. This will only give rise to difficulties if the same pattern is produced when two different characters are intended; there is no difficulty in dealing with a number of different patterns which denote the same key. I cannot say yet whether or not this problem is likely to be significant, because I have not been able to carry out the necessary experiments; informal observations suggest that such confusion will occasionally occur, but not sufficiently often to offset the benefits of easier keyboard access. Clearly, much depends on the abilities of the typist.

The second proposal[2, 3] embodies the use of combinations of adjacent keys which can be pressed simultaneously with one finger. With some variation over the keyboard, each of the conventional keys can be associated with several possible key groups, as illustrated in the table below. The key dimensions and layout are again taken from a real keyboard; the grey ellipse represents the finger position in each case, and (except in the case of the quadruplet) is somewhat smaller than the key contact made by my index finger. The quadruplet is rather hard to manage, but the corresponding combination on the row above that illustrated (the group { A, S, Q, W }) is much more readily accessible.



If all these combinations could be used for every conventional key on the keyboard, the result would be in effect to make available seven times as many keys as are in fact present. (More precisely, there would be seven times as many spatial positions which could be detected independently, and could be associated with different meanings.)

Practically, the factor is likely to fall well short of seven, but a significant increase is still possible. The additional positions can be preprogrammed with useful meanings, or left undefined, with provision for programming them to denote any arbitrary text sequence.

MULTIPLE KEYING EXPLORED : A PROBLEM.

In attempting a practical implementation of these ideas, one considerable obstacle is imposed by the architecture of the computer itself. I mentioned earlier that there were obstacles in the way of identifying just which keys were down when a large number were pressed, and now it is appropriate to return to that topic.

Once the keys have been pressed, the function of a computer keyboard is to present the keyboard signal to the attached computer by transforming that signal into an electrically encoded equivalent. Perhaps the most primitive way of forming the attachment would be to use a separate connecting wire for each key, so that the computer could inspect the state of any key at any time as required by its programme. In practice, this would not be at all convenient, and it is usual to use a rather different sort of connection, which nevertheless gives the computer the information it needs in order to work out which keys are pressed at any moment. As the computer can no longer inspect the keyboard directly, though, the keyboard must tell the computer every time its state changes, so that the computer can keep track of what's going on. The keyboard therefore sends a message to the computer every time a key is pressed, and every time a key is released. Each if these messages must contain a number identifying the key which has moved (the keycode) and an indication of whether the movement was down or up.

Given this information, it is comparatively straightforward to keep track of what is happening at the keyboard – provided that the information is correct. Unfortunately, with common techniques used to manufacture keyboards, it is less than easy to ensure that the information remains correct as more and more keys are depressed[4], so in practice a keyboard might only report a limited number of key-down signals before some keys are released. This is termed *rollover*. The rollover limit for a keyboard depends on its architecture; for the keyboard on the portable computer used for the development work, the rollover limit is two. I wanted to continue working with this computer so that I could guarantee compatibility between different people's performances, so I was limited to detecting a maximum of two simultaneous key depressions.

This is clearly a serious disadvantage so far as the first proposal is concerned, so I have made no progress with that line of research. For the second proposal, though, the damage is significantly less severe; the triplets and quadruplet are not accessible, but the three pairs remain, and offer a potential fourfold "expansion" of the keyboard. This is the work reported in the next section and what follows.

The first proposal, though in difficulties, might not yet be dead. Though the rollover limit filters out some signals, they are still produced by the keyboard, so if it were possible to circumvent the component which applies the rollover filter it might still be possible to acquire enough information about the signal applied to the keyboard to continue with the investigation. Two possibilities (not necessarily a complete list) appear :

- Burrow deeper into the system, and acquire the signals closer to the keyboard. This might be possible. After a protracted exchange of views with Compaq support personnel[5], and reference to various Compaq documents[6], it seems likely that this approach would be possible[9]. I have not pursued this possibility further, partly because of time limitations but also because any such solution is likely to be limited to Compaq machines – and, in view of differences between models[6], possibly even to an individual model. It might still be worth following up to give a testbed for exploratory research on the first proposal, but falls a long way short of an effective widespread solution.

- Use a separate keyboard. Keyboards not subject to the potential confusion addressed by rollover techniques can be built, though they are a little more complicated – and, therefore, more expensive – than those of simpler architectures. If a separate keyboard which will provide all the required key signals can be found, attach it to the computer though an external port and write the software for that keyboard. The drawback of this solution is that it is more expensive, and we no longer have a method requiring no additional hardware.

I have not yet decided what to do about this problem. Having spent quite a lot of time getting this far, it seemed better to use the remaining time to follow up the second proposal, for which some progress was possible. Meanwhile, the first proposal is in abeyance.

THE SOFTWARE.

The software is a programme written in Borland C++[7] and compiled for Windows 3.0. With the intention of making the programme as portable as possible, the programme itself contains no features dependent on Windows, and should compile satisfactorily with little or no change using a C compiler for MS-DOS. Some MS-DOS feature are used, so the programme cannot be transported to other systems without some change. Operating instructions have been written[8]; a brief description follows.

The programme has two major modes. It begins in a setting-up mode, where it runs as a conventional programme and various parameters can be set. In use, it is in testing mode, and terminal input interrupts are intercepted; standard Windows actions are unlikely to operate, and the mouse is dead. The programme in testing mode can be run either as a sticky-key system or with multiple keys as described earlier; the intention is to facilitate easy comparison between the methods.

In testing mode, key-down and key-up interrupts are intercepted by the software and used to maintain an internal record of the state of the keyboard at all times. Details of what happens thereafter depend on the sort of keyboard usage selected.

When using multiple keys, a cycle of activity begins with a key-down interrupt. A timer (implemented as a busy wait) is started, and all keys depressed at any time during the set interval remembered, even if they are released before the interval ends. At the end of the interval, the set of keys which were depressed during the interval are presented to the following software as a *keyset* assumed to have been pressed simultaneously. Notice that this part of the system will operate whatever the rollover restrictions. The keyset is converted into text by reference to tables. These are restricted to individual keys or key pairs, so in effect assume that the rollover limit is 2. The text retrieved is the standard lower case character if one is defined for the key, the upper case character if a key is pressed together with its right-hand neighbour, again if defined, and arbitrary stored text for any other keyset. A keyset for which no value is stored is signalled by a beep.

When using sticky keys, no time delay is imposed, but a depression of the shift key is remembered until another key is pressed; the second key is then regarded as upper case. This convention gives access to upper case characters without requiring simultaneous depression of two potentially widely separated keys. Caps lock operation is implemented as shift lock. As with the multiple key method, text can be defined for keys not otherwise used, though fewer combinations are available; function keys and "upper case" function keys are the easiest to use.

In both cases, text produced is normally displayed on the screen. There is no provision for saving the text as a file, as the programme is purely for exploration. (With hindsight, it might be better to save the text, if only to give experimenters some incentive to try out the system reasonably extensively.) So far as possible, the two usage methods are kept compatible, so text defined with sticky keys can be retrieved using corresponding multiple key combinations.

There is provision for monitoring the keyboard use. The system can display the detected key-down and key-up signals as well as the character equivalents; this output can be directed to the screen or to a disc file. In normal use, one would probably direct the monitor output to a disc file while presenting the ordinary text on the screen.

THE INVESTIGATION.

Because of time constraints, no formal evaluation of the system has yet been possible. The informal comments which follow are based partly on my own experience with the system, but more on some trials carried out with the very valuable help of Rachael Gascoigne. Rachael has cerebral palsy, because of which her hand movements are constrained; she had significant previous experience of using computers, which was an advantage in understanding the nature of the experiment.

The trials began with the notebook computer used in the system development, but the small flat keyboard proved difficult for Rachael to manage. In subsequent sessions, we therefore used a conventional desktop machine with a larger keyboard. This shift was valuable, because it showed up the importance of the keyboard geometry on the system's performance. The pattern of ease of use of key combinations using the larger and more steeply banked desktop keyboard was different from the corresponding pattern for the notebook keyboard; the horizontal pairs (Z and X in the diagram above) were significantly easier to use than the other combinations (Z and A, and Z and S). This caused no particular difficulty in the trials, as we did not experiment with other than horizontal pairs, but might be a limitation in more ambitious attempts.

The key spacing on the desktop machine was also the source of some difficulty. The keys were more widely spaced than those of the notebook machine, and the gaps between the heads of the keys were comparable in width to Rachael's fingers, which made it difficult for her to press two keys simultaneously. This problem was solved by the use of a rubber thimble, which effectively increased the diameter of Rachael's finger and solved the problem.

The value chosen for the time delay within which keypresses were considered to be simultaneous turned out to be important. As Rachael's typing speed was not large, a rather long delay could be chosen, giving her time to ensure that both chosen keys were pressed within the time limit and without significantly slowing down her overall speed. This mode of operation is in effect a variation on the original expectation, in which "sliding" movements from one key to an adjacent key replace the original intention of simultaneous key depressions. This might open a way to the reinstatement of the key combinations which were harder to use with the desktop keyboard.

DOES IT WORK ?

In assessing the value of the method, the most encouraging sign is Rachael's opinion that it is worthwhile. While she was not immediately able to press the combined keys, she believe that it would be possible with practice, and worth the effort.

Rachael's opinion was derived from experience limited to the use of the upper case characters preprogrammed as the meanings of the horizontal key combinations, so does not take into account any benefit from the use of a wider range of key combinations. Even at this level, though, it can be argued that the multiple key method is twice as good as sticky keys : an isolated upper case character requires

three keystrokes on an unmodified keyboard (caps lock; character; caps lock), and two with sticky keys (shift; character), but only one with multiple keys (character and neighbouring key).

A further difficulty became evident in my own experiments with defining character sequences for unused key combinations : after saving even fewer than ten sequences, it was not at all easy to remember the key combination used for a stored text. The task of choosing a key combination to match a text is not made any easier by the restriction of combinations to pairs which happen to be available on the keyboard, so preferred abbreviations cannot usually be used. While lists can be printed out when required, some more direct means of finding the code for a stored sequence is desirable.

On the basis of these experiments, I judge that the method is worthy of further investigation. Significant improvements to the present implementation are possible, and more extensive controlled experiments are desirable.

ACKNOWLEDGMENTS.

It is a pleasure to thank Rachael Gascoigne, Alison Pearce, and Jane Smedley, of York College of Further and Higher Education, for their generous and friendly help.

REFERENCES.

1 : G.A. Creak : "Notes for a seminar : Insights from a System Specification Aid", *Sigcaph Newsletter*, in press.

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

3 : G.A. Creak : *Faster communication through multiple key operations*, unpublished Working Note AC98, (May, 1996).

4 : J. Fulcher : *An introduction to microcomputer systems* (Addison-Wesley, 1989), page 104.

5 : Electronic mail exchanges with Andy Switzer (Compaq customer technical support, USA) and Andrew Jury (Systems support, Compaq Europe – more helpful), April - October, 1996.

6 : *Technical Reference Guide for the Compaq LTE Elite family of personal computers* (Compaq, 1994); *Technical Reference Guide for the Compaq Deskpro 486/33L personal computer* (Compaq, ?).

7 : *Turbo C++ 3.0 for Windows* (Borland International Inc, 1991).

8 : G.A. Creak : *Instructions for the multiple key interface*, unpublished Working Note AC109, (December, 1996).

9 : G.A. Creak : *Subverting BIOS for the Compaq Elite*, unpublished Working Note AC108, (December, 1996).