Computers Educ. Vol. 14, No. 4, pp. 335–341, 1990
Printed in Great Britain

0360-1315/90 $3.00 + 0.00
Pergamon Press plc

AN EXTENDED USER INTERFACE FOR CAL SYSTEMS

DAVID DAVENPORT
Department of Computer Engineering and Information Sciences, Bilkent University, P.O. Box 8, 06572 Maltepe, Ankara, Turkey (Email address: DAVID@TRBILUN on EARN/BITNET)

(*Received* 13 *July* 1989)

Abstract—This paper examines the problems of complexity in human–computer interaction from the perspective of CAL systems. By considering the very special demands of young, pre-reading age children, it shows how pointing type interfaces can be extended in a simple and natural manner based on the notion that the position of, and spatial relationship between objects, is of significance. Various examples based on this extended interface concept are introduced to demonstrate its functionality.

INTRODUCTION

The idea of CAL has been around for a long time; for an excellent history/analysis of the field see Solomon[1], while for a more detailed practical approach, the book by Barker and Yeates[2] is recommended. Only comparatively recently, however, has really powerful hardware become cheap enough to be readily accessible for CAL purposes. Besides its affordability, the other distinguishing feature of such equipment is its graphics capability, which opens the way to interesting, varied and sophisticated material presentations, something previously very difficult to attain. Unfortunately this very sophistication can also lead to problems; there are several reasons for this, most of which centre around the complexity of the system as perceived by its users, both students and teachers.

Often software designers, in attempting to utilize all the available features of a machine, can unintentionally overload the user's ability to manage it efficiently. This situation is already very apparent in industry/commerce where it is not unusual to find that a program comes complete with (and requires), a glossy 500 page manual detailing how to use its every feature. Typically even adult workers in a company are unable or unwilling to read such an epic, with the result that the software stands idle or is used inefficiently. Such a situation is deplorable from an educational viewpoint, and even worse if it occurs within an educational setting where the objective is to learn the package's contents, not its use! As an extreme, consider that no self respecting 3 year old is going to invest the time necessary to digest such a manual; it may take another 5 years, after which, learning that one orange plus one orange gives two oranges would probably not be too rewarding!

Quite simply then a lot of software is too difficult, or at least too time-consuming, to learn to use. The problems are further compounded when numerous different programs are employed within a school, as may be necessary to provide adequate coverage of course material. This can also occur by accident when, as is often the case, equipment is placed into schools with little or no readily available software. The ensuing scramble to find or produce some sort of learning material often results in a somewhat haphazard collection of programs, each of which has a different set of user controls. With no standardization, each package requires to be learnt anew, the poor student being forced to perform mental contortions simply to be able to use the packages, let alone learn anything new from them.

Admittedly the situation is not always as bad as portrayed here. The importance of a good user interface, the communication boundary between computer and user, to any human–machine interaction is being increasingly recognized in CAL, as in other areas of computing[3]. The trend is towards more logical, uniform, simple to use systems. Luckily the sophistication which causes many of the problems in the first place can also aid in providing solutions. This process is already very apparent in the window/mouse type interfaces, which attempt to provide just such an all-encompassing yet simple environment. (For an example of such a system see [4].)

The present work serves to extend this concept. It starts by looking briefly at some of the reasons for complexity in user interfaces and then introduces an experimental user interface developed as a result of these observations to investigate the problems of pupil/computer interaction. Several examples based on this interface are then presented, demonstrating its ability to produce material suitable for students of all age groups.

COMPLEXITY IN USER INTERFACES

User interfaces are bounded by the available input/output devices. Originally this meant teletypes operating at 10 characters per second, severely limiting communication possibilities. These were gradually replaced with VDUs which, while much faster, were still limited to the same textual based character set. The 'soft' (temporary) nature of their display allowed considerably improved information presentation, however the input device remained, and still generally remains, the typewriter-like keyboard. Over the years keyboards have changed very little except for a considerable growth in size. Nowadays it is not uncommon for them to have over 100 keys, many of these multi-function, providing great flexibility but at the price of a corresponding increase in complexity. Indeed it is this sheer volume of keys which gives rise to many of the problems of standardization in user interfaces. There are simply too many keys for designers to choose from, with the result that every software house assigns different functions to different keys. As programs have grown in sophistication, the number of options available to users has increased, and this has become more and more of a problem, culminating in the 500 page manual syndrome mentioned above.

Of course this is not to deny that the keyboard is of great value; quite the contrary, its continued existence bears tribute to its utility. If the user is expected to enter a lot of text as quickly as possible, then the QWERTY keyboard is certainly a reasonable choice. In CAL, however, this is rarely the case for several reasons. Firstly, the computer is generally incapable of analysing more than a few words of free form text, so that most student responses are purposefully limited; secondly, many potential users, e.g. 2 year olds, are as yet unable to cope with its complexities; and thirdly, even those who can understand it find it a very unnatural, slow and clumsy device unless practised frequently. Such difficulties have led some researchers to suggest alternative layouts which reduce learning time and/or improve data entry speed[5,6]. Others have suggested so-called concept-keyboards, which do not allow full alpha-numeric data entry, but rather, offer only a limited number of keys each of which can be dedicated to a specific application function[7].

Despite this, however, today's microcomputers still retain the standard QWERTY keyboard as the primary input device and the VDU like screen as output device. The display has generally been enhanced with sophisticated bit-mapped graphics, allowing the display of non-textual material such as graphs, charts and pictures. This has brought with it an effective increase in output bandwidth (evidenced by the expression, "a picture is worth a thousand words"), and a corresponding improvement in user acceptance. It has also brought with it a renewed interest in pointing devices for input. Pointing as a means of identification/input is much more natural than typing, understandable even to 2 year olds, however it obviously has limitations as far as text entry is concerned. Within the limited entry required by CAL, this problem can be alleviated by multiple choice questions, by "soft" keyboards or by combined use of pointing device and keyboard as appropriate.

One of the first commercially available pointing devices was the light pen. Unfortunately arm fatigue resulting from continually having to hold the pen to the screen, combined with the fact that many early examples proved somewhat unreliable, resulted in poor user acceptance. The touch-screen, where one literally places one's finger on the desired screen location, has had a more lasting impact; however, it is generally limited to low resolution applications, such as selecting one of a limited number of options. For more detailed work, the tracker-ball—a sort of upside-down mouse which remains static on the table, the user moving the ball with finger tips—has long been popular in design environments, although it shares many of the problems of its successor, the mouse, in terms of usability.

Both the mouse and the tracker-ball overcome the problem of arm fatigue by remaining on the desktop, although the mouse must be physically moved across the surface requiring an arm/wrist action rather than just finger tips. In comparison with the light pen it does, however, have other defects. One difficulty arises from the fact that one does not physically point to the things that one wishes to identify. Even more serious is the fact that the mouse is an indirect pointing device, in other words, the movement and position of the mouse itself does not correspond one-to-one with that of the cursor on the screen. This discrepancy between hand and eye, particularly noticeable if the mouse is held and moved at an angle (as is natural!), is somewhat disconcerting and can take considerable effort to master. In very young children, who are only just becoming accomplished with conventional co-ordination tasks, this can entail a severe cognitive learning penalty. Despite this, however, the style of interfaces generally associated with the mouse, the so called WIMP (window, icon, mouse, pointing) interfaces, are gaining considerable popularity. The single/twin button control and pop-up explanatory menus, offered by such interfaces are a far cry from the multiple unlabelled function/control keys of the conventional keyboard, although it should perhaps be noted that much improvement could be achieved without the use of the mouse.

Not surprisingly, then, the mouse is readily accepted even by naive users, who previously refused to touch the computer for fear of the keyboard. One problem with the window/mouse interfaces remains, though, and that is the fact that they generally rely on the user being able to read and understand the contents of pop-up menus and the like, prior to pointing to the desired selection. While very young children can point, they cannot read, so the interface can appear extremely confusing to them. Icons, pictographic representations of data or processes within a computer system[8] are a step towards these very young users, but only a small one. The design presented below effectively extends the concept of an icon so that it becomes familiar enough for them to understand and use.

INTERFACE DESIGN, OBJECTIVES AND PHILOSOPHY

The major objective was to produce a simple natural interface which could be used by a wide range of pupils. These would include pre-reading 2 year olds, who would use the machine to learn pattern recognition, letters, words, numbers and counting in order to develop their skills before reaching school, through teenagers who, while fluent in their mother-tongue are now struggling to master the complexities of a second language, to university undergraduates and industrial workers, who must acquire new and ever more complex skills but to whom reading and writing is no problem. Obviously, besides a vast range of abilities there is also a significant variation in material to be learnt. Another objective of the interface was therefore to investigate whether a single conceptual structure could support such a huge range.

It is perhaps not surprising that the youngest users present the biggest problems and challenges. Their inability to read severely restricts the form the interface may take. On the other hand, if it can be designed simple enough for them to use, then older pupils should have no difficulty. While small children cannot read, they can point, so the window/mouse style interface seems a natural starting point. As a means of interacting with the world, however, pointing is somewhat limited. Youngsters quickly progress from pointing to holding, moving and rearranging objects, and it is this concept that the interface described here attempts to embody. By making graphics, letters, words, even whole sentences, "graspable" with the mouse and allowing the user to pick them up and move them around the screen world, rearranging them as desired, it provides a very sophisticated, powerful yet natural environment in which to interact. In effect, it extends the existing window/mouse interfaces into a dimension more akin to real world objects, but with the added advantage of electronic imagery.

This new world of electronic objects can be made to resemble very closely the children's home/school environment and the sorts of toys and games that are employed there. Most of these relate to objects being collected into associated groups or placed in some specified relationship with each other. Teaching the child to recognize shapes in various guises is one aspect of this, which obviously widens to include numbers and letters too. They can be viewed as shapes on cards, which must then be placed in appropriate spatial relation with respect to each other. Relating words to pictures is yet another aspect. Not surprisingly this concept extends further, through sentences and lists of instructions, which must be placed in correct sequence, to more familiar objects such as pieces of paper, electronic devices and the like. We live in a world governed by spatial relationships; this extended user interface tries to capture more of the flavour of that world.

The interface itself is very simple, consisting of only three conceptual entities; buttons, menus and objects.

Buttons and menus

These are static entities which serve to invoke some system function. They behave exactly the same as their counterparts in existing systems[9]; however, they need not be purely textual, but can also display any sort of graphic as required. This allows such entities to be understood and used even by non-readers.

Objects

Objects may also display both text and graphics. Unlike buttons and menus, they are potentially movable. To move an object one simply points to it and presses the left mouse button, then drags it wherever one wishes. Releasing the button 'drops' the object. Objects are generally opaque and move over one another; however translucent objects are also possible. Objects can also possess other properties akin to buttons and menus. As an example, pointing to an object and pressing the right mouse button may display additional information attached to that object. If the object is a letter in a child's game, then this may be a picture of an article beginning with that letter; if it is a word in a foreign language tutor, it may be a menu giving options such as example usage, a graphic if appropriate, synonyms, meanings etc. Since the object's position on the screen may be important, the interface includes special functions to determine if one object is left, right, above, below, touching or completely overlapping another object.

The following examples should serve to give some idea of the potential of such an interface.

EXAMPLE APPLICATIONS

SPELLER

SPELLER is a simple program designed for young children of 2–3 years who are learning to recognize letters and words and to relate them to objects. In its most basic form, it selects one word

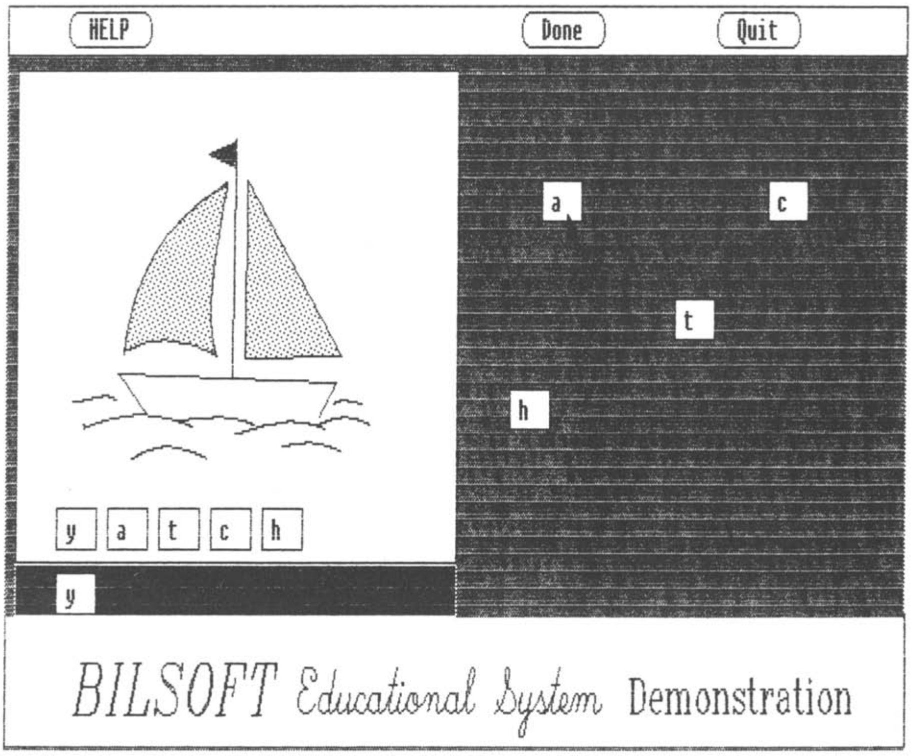
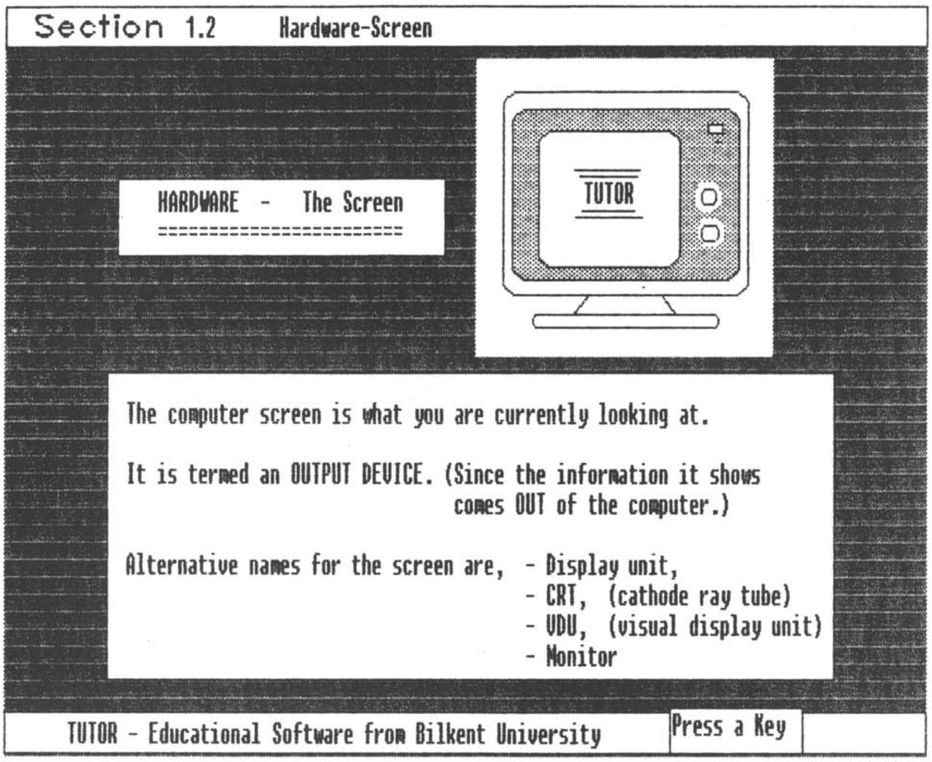


Fig. 1. SPELLER.



Section 1.2 Hardware-Screen

HARDWARE - The Screen
========================

The computer screen is what you are currently looking at.

It is termed an OUTPUT DEVICE. (Since the information it shows comes OUT of the computer.)

Alternative names for the screen are, - Display unit,
- CRT, (cathode ray tube)
- VDU, (visual display unit)
- Monitor

TUTOR - Educational Software from Bilkent University

Fig. 2. TUTOR.

at a time from a teacher supplied list, displays a corresponding picture (if one is available), writes the word itself under it, and duplicates the letters of the word at random in another section of the display (see Fig. 1). Using the mouse the pupil must "pick-up" the duplicate letters and place them under the corresponding ones in the correctly displayed pattern. Appropriate feedback is provided by both sound and animation.

As the child becomes more and more proficient, the "game" can be extended to force them to place the letters in their positions in the correct order. It can also add extra letters from which to choose, ultimately building up to a complete alphabet. Another variation is to start omitting letters from the pattern, so that the child has to remember first some, then finally all the letters of the word. Alternatively, it may display the word and a selection of graphics, the student having to select the one matching the word. Further extensions are possible, for example, displaying the picture and a number of similarly spelt or sounding words and asking the pupil to select the correct one. All these possibilities can be easily accommodated within the same basic program and using the same user interface.

TUTOR

TUTOR is a less sophisticated but more general purpose CAL tool. To the user it appears like a conventional frame based CAL system (however, it embodies several new ideas which are beyond the scope of this paper, see[10] for details). It reads information from a teacher supplied file and displays it for the student to absorb. Material, both text and graphics, is divided into blocks, each block being displayed in a separate window on the screen (see Fig. 2). The file is read like a script and determines when windows are to be displayed, how they should be moved, hidden, or erased, providing a powerful yet simple to use presentation mechanism. The file also contains questions which may be interspersed with the display material to check student understanding. These too get placed in windows. A variety of question forms may be included which use either keyboard or mouse as input mechanism. These can range from conventional text based pattern matching types used with keyboard entries, to more sophisticated mouse driven questions similar in structure to the word games of SPELLER described above.

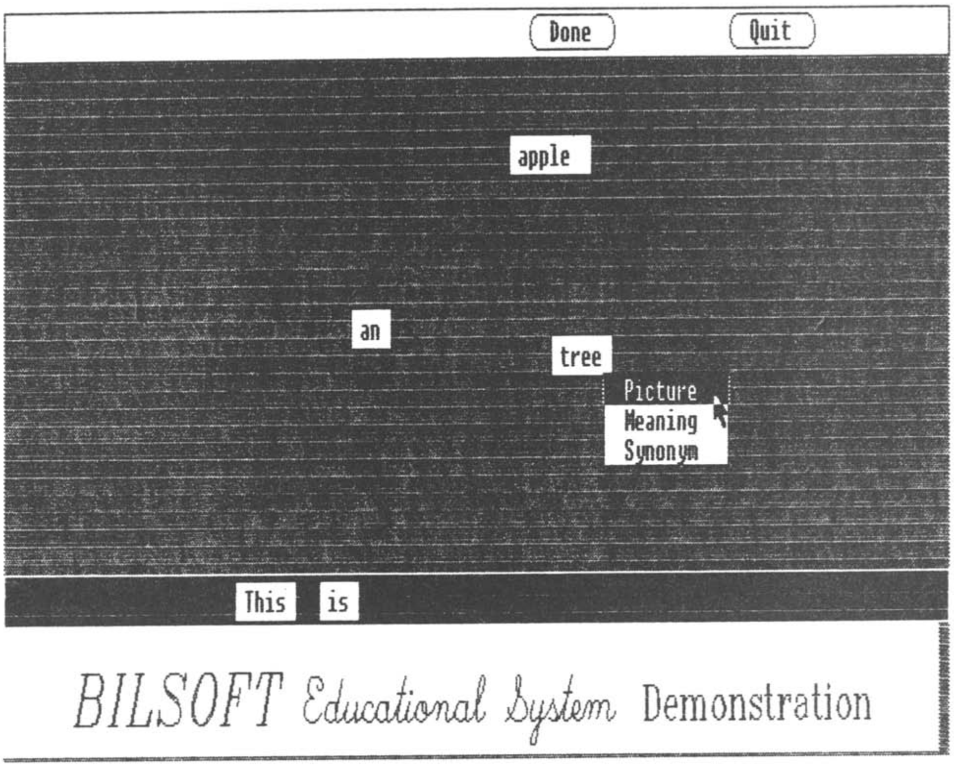


Fig. 3. EDU.

Other examples

Many other possibilities exist. For example, Fig. 3 shows a screen from an experimental language tutor, which in essence is very similar to SPELLER, except that it treats whole words as objects, rather than the individual letters. The figure also shows how additional information about a specific object may be obtained as mentioned earlier. Another possibility is a digital simulator. Symbolic electronic components can be treated as objects, being picked from a menu (where they are displayed as graphical symbols, not text), and then placed wherever desired. Pointing allows connection paths to be made, and then simulation can proceed by pressing an appropriate button. Yet another option is a program editor. Statements can be picked from a menu and appropriately placed. Variables can be entered either from the normal keyboard or from a pop-up 'soft' keyboard. Once entered they become available on a menu, thus overcoming the problem of misspelling. The same basic concept may be further extended to cover all the basic editing operations.

SUMMARY

A user interface has been presented which extends the pointing, WIMP, interface into the realm of objects which can be picked up and rearranged. The concept is simple and natural enough to be grasped even by very young, pre-reading age children, yet powerful enough to provide a serious work enviroment. Several applications based on this interface were introduced. The first of these, SPELLER, has not yet undergone extensive trials, but has been easily and cheerfully practiced by the children who have played with it. On the other hand, TUTOR has been successfully used to teach many hundreds of students the basics of computers and the MSDOS operating system as a preliminary to their undergraduate programming classes. While a lot of work remains to be done, it is hoped that these examples demonstrate the potential of the basic concept as an extension to existing user interface designs. In the author's opinion we are gradually moving to the situation where the keyboard fades in importance and we are left with only a bare pointing device to manipulate, the concepts introduced here represent a small but vital part of that transition.

Acknowledgements—The author gratefully acknowledges the assistance of Dr Altay Güvenir, Dr Güneş Davenport, and especially of Ferhat Büyükkökten, who undertook most of the programming effort.

REFERENCES

1. Solomon C. *Computer Environments for Children, a Reflection on Theories of Learning and Education.* MIT Press, Cambridge, Mass. (1986).
2. Barker P. and Yeates H., *Introducing Computer Assisted Learning.* Prentice-Hall Int., New York (1985).
3. Nievergelt J., Ventura A. and Hinterberger H., *Interactive Computer Programs for Education: Philosophy, Techniques and Examples.* Addison-Wesley, Reading, Mass. (1986).
4. O'Brien B., *Opening Windows.* Scott Foresman, London, (1987).
5. Olson D. W. and Jasinski L. E., Keyboard efficiency. *Byte* **11,** 241-244 (1986).
6. Roberts M. and Rahbari H., A multipurpose system for alpha-numeric input to computers via a reduced keyboard. *Int. J. Man-Mach. Stud.* **24,** 659 (1986).
7. Adams J. (Ed.), *Learning to Cope—Computers in Special Education.* EMAP, London (1983).
8. Gittens D., Icon-based human-computer interaction. *Int. J. Man-Mach. Stud.* **24,** 519-543 (1986).
9. Williams G., Hypercard. *Byte* **12,** No. 14, 109-117 (1987).
10. Davenport D. and Güvenir H. A., TutoR: an experiment in computer-aided-learning. *Proceedings of The Fourth International Symposium on Computer and Information Sciences,* Cesme, Izmir, Turkey (1989).