

MUESLI NEWS

October 1985

Editor

David Eastment

Contents

CALLing Programmers - Chris Jones
Talking Basic
Making your own cursor on the Beeb
Ifs can be avoided - true or false?
Who CALLs the tune? John Morgan, Pilgrims
CALL Software
Text reconstruction programs
Crossword

Muesli News

Issue Number 3: October 1985

Editorial: Welcome to the third issue of the MUESLI Newsletter. I hope the new typeface is not too much of a strain on the eyes: it does enable us to pack much more in! My apologies to everyone for having taken longer than expected to produce this issue. We hope that future issues, no matter how slim will appear every two months at least, so please keep the letters (even the nagging ones!) rolling in.

This issue contains news of MUESLI's first AGM; a report on our last meeting; and an outline from John Morgan on the current quality of EFL software; hints and tips; and last, but by no means least, a crossword from BUFFER. The December issue will carry a special section on networking, a review of Wordstore, and more software listings.

Meeting and AGM, May 1985

A meeting of MUESLI was held on 18 May 1985 at Davies School, Eccleston Square, London and was attended by 17 members and guests.

The invited speakers were John Allen and Tony Hufton, of MAST Learning Systems, a company which produces CBT materials for industry. Between them, they gave a lively demonstration of Mastwriter, an AECAL authoring system. This system was shown during a variety of software packages on a Tandberg TCCR530 and an IBM PC (though versions of Mastwriter are available for other hardware configurations). A picture-dictation activity, produced by Tony Hufton, was particularly relevant to ELT needs.

After lunch, the AGM took place. The minutes were as follows:

11 members attended.

Glyn Jones was asked to chair the meeting, Nick Reckert to minute.

The Chairman proposed, and the meeting accepted that voting would be on the basis of one vote per non-affiliated member or per autonomous institution or per branch.

David Eastment, who has carried most committee activities since the formation of MUESLI, delivered his report. The organization had grown, in 18 months, from 12 to 40 members and had averaged a meeting every three months. The accounts (circulated) showed a healthy bank balance of 185.96. Speakers fees had averaged 25.39. Secretarial costs had been lightened by the generous assistance of Bell College Saffron Walden. He appealed for such aid, from member schools and publishers, to continue. The meeting thanked David for his hard work.

It was agreed to maintain membership subscriptions at the rate of 10.00 per annum.

The following formal aims of MUESLI were agreed: To promote the use of microcomputers in EFL by:

providing a forum for the exchange of ideas and information among current users of micros in ELT;

providing information for individuals and institutions to develop the use of micros in ELT;

providing a link between users and producers of commercial CALL software;

maintaining contacts with developments in related fields.

The following officers were elected:

Secretary/Chair: Glyn Jones (Eurocentres)

Newsletter Editor: David Eastment (Bell)

Information Officer/Librarian: Brian North (Eurocentres)

Treasurer/Membership Secretary: Nick Reckert (CUP)

Activities/meetings for the following year were discussed at some length.

The day ended with hands-on software trials.

(Nick Reckert)

Future Meetings

At a meeting held at Cambridge Eurocentres on 17th October 1985, the following schedule for meetings was announced:

December 8th 1985: John Higgins and Alison Piper (Bell College Saffron Walden)

January 26th 1986: Networks and Videotex (Bell College Saffron Walden)

March 1986: Artificial Intelligence (Davies School, Eccleston Square)

Post-IATEFL: Interactive Video (Venue to be announced)

May/June: Real applications for micros (Cambridge Eurocentres)

More details of these events will be available at the December meeting. In addition to the above, MUESLI has been asked to organize a CALL day at the 1986 IATEFL, along the lines of this years event.

(de)

CALLing Programmers

Starting with this edition, the Newsletter will have a regular section on CALL programming. The feature will focus mainly on the BBCB, but were obviously interested in useful ideas for other widely used machines.

Hints and tips from members are welcome, and should be sent to Chris Jones, 40 Mortlake Road, Kew, Richmond, Surrey TW9 4AT.

Meanwhile we kick off with news of a book on programming, a cursory glance at cursors on the BEEB, and the truth about truth and falsity.

TALKING BASIC

Graham Davies Talking Basic has now been published by Cassells. The book takes a detailed look at BASIC, mainly on the BBC and Commodore 64, and is aimed specifically at CALL: programs which draw sine waves and calculate the number of gallons of paint required to redecorate a living room are not included. Instead, readers are shown how to compare strings, convert from upper to lower case, crunch out unwanted spaces, sort lists, handle files, keep scores, create foreign characters, and a lot more besides.

The book starts from scratch (its 340 pages long), and at first glance looks well written, user-friendly and comprehensive. Available from all good bookshops at 6.95.

MAKE YOUR OWN CURSOR ON THE BEEB

Youve probably noticed a number of different cursor shapes on the BEEB: chunky in Mode 6, thin Mode 7, thicker in Wordwise, and that nice iwhole character cursor you get in editing mode.

It is in fact quite simple to program the cursor to appear as you want, by using two VDU 23 commands. The two commands control the starting height of the cursor and its finishing height. The height numbers range from 0 to 11 in Modes 0-6 and from 0-19 in Mode 7.

Finishing height first. The command `VDU23,0,11,FH;0;0;0;` - where FH is your finishing height number (in Mode 7 its normally 19).

The starting height is more intriguing. Its `VDU,0,10,(96+SH);0;0;0;` - where SH is your starting height (normally 18 in Mode 7). The intriguing bit is the 96, because this number also controls the flash rate of the cursor. But first the cursor shape:

To get a whole character cursor in any Mode, leave the finishing height alone, and start your cursor at the highest possible point (which is 0). $96 + 0$ is 96, so your command is `VDU23,0,10,96;0;0;0;`. The command to get it back to normal depends on your Mode. In Mode 7 you want $96+18$, so type `VDU23,0,10,114;0;0;0;`. In Modes 3 and 6, substitute 105 and in all other Modes type 104. By playing with the numbers, you can also create cursors that appear half way up the character, or even above it.

And now for the cursor flash. Heres what you get by varying the 96:

96 cursor flashes slowly (default value)
64 cursor flashes fast (as in Wordwise editing mode)
32 cursor disappears altogether
0 cursor doesnt flash at all.

By adding whatever starting height you want to any of these numbers, you can control both the shape and behaviour of the cursor.

Ive found no use for more exotic cursors, but a larger-than-normal cursor in Mode 7 can be very useful when want to be sure the student knows where (s)he is on screen.

IFs CAN BE AVOIDED: TRUE OR FALSE?

An underexploited fact about computers is that they assign values to truth and falsity. If youre not sure what this means, switch on and type the following:

`X=4 <RET>` followed by
`PRINT X=4 <RET>` - the computer will respond by printing -1,
Now try `PRINT X=3 <RET>` - this time the computer responds with 0.

Once youve given X a value (in this case, 4), you can ask the computer things about X. `PRINT X=4` means, in effect Tell me if $X=4$ or not. The same goes for `PRINT X=3`. You will have gathered, then, that the BBC gives the true value of -1, and falsity the value of 0. Since X does equal 4, the first statement is true, and is worth -1. Asking about any other value for X will produce the answer 0.

You can do the same with strings. If `A$=dog`, then `PRINT A$=cat` will produce 0, while `PRINT LEFT$(A$,2)=do` will get you -1, as will `PRINT A$=dog` and `PRINT LEN(A$)<4`. The feature can be used within programs, and is particularly useful as an alternative to IF statements. Supposing you have 10 different strings - `A$(1)` to `A$(10)` - and you want to know if the students input (`G$`) is the same as any of them. This would do it:

```
100 Y% = 0:FOR X% = 1 TO 10: IF G$ = A$(X%) Y% = 1
110 NEXT
```

Using truth/falsity values, we can achieve the same in one line:

```
100 Y% = 0: FOR X% = 1 TO 10: Y% = Y% - (G$ = A$(X%)): NEXT
```

In the first version, Y% comes out as 1 if a match is found. What about the second version? Essentially it works in the same way: for each value of X%, the computer compares G\$ with A\$(X%) and comes out with a value: 0 if they don't match, -1 if they do. It then adjusts the value of Y%. If no match is found, Y% doesn't change (since Y% - 0 = Y%). But if a match is found, Y% is incremented by one: Y% - (-1) is the same as Y% + 1.

Truth and falsity really comes into its own when there are a number of different IFs. Suppose we want to allow a user to move the cursor around a particular area of screen using the arrow keys, and with wraparound - that is, if the cursor goes off one edge it reappears at the opposite edge. First we disable the editing function of the arrows by typing *FX4,1, and set up H% and V% to determine the cursor's horizontal and vertical position. Say we want the cursor to be confined to a square starting at 5,10 and ending at 34,19. With IF statements, we might end up with something like this:

```
100 H%=f:V%=10                Set up starting position
110 VDU31,H%,V%                Position cursor at H%,V%
120 REPEAT:Y%=GET:UNTILY%>135 AND Y%<140 Get arrow character
130 IF Y%=136 H%=H%-1:IF H%<5 H%=H%+30If left, decrease H%
140 IF Y%=137 H%=H%+1:IFH%>34 H%=H%-30    If right, increase H%
150 IF Y%=138 V%=V%+1:IF V%>19 V%=V%-10    If down, increase V%
160 IF Y%=139 V%=V%-1:IF V%<10 V%=V%+10    If up, decrease V%
170 GOTO 110                    Go back for more
```

Truth/falsity enables us to condense lines 130-170 into one line, and in the process lose all eight IF statements. For clarity's sake, however, we'll split the process into two lines:

```
130 H%=H%+(Y%=136)-(Y%=137): V%=V%-(Y%=138)+(Y%=139)
140 H%=H%-(H%<5)*30+(H%>34)*30: V%=V%+(V%>19)*10-(V%<10)*10:GOTO 110
```

The procedure is not complicated. Of the four possible values for Y%, only one can be true after any one GET. The untrue statements in line 130 generate the value 0, and so have no effect: the true one, however, increases or decreases H% or V% by one, as required. The programmer only needs to get the plus and minus signs right.

Line 140 deals with the wraparound, and makes use of multiplication. The first part of the line, for instance, checks whether H% has fallen below 5: if it hasn't, the computer subtracts 0*30 from H%, that is, nothing. If it has, however, the statement is true, and -1*30 is subtracted from H% (ie 30 is added), taking the cursor to the right edge of the square. And the rest of line 140 works in the same way.

One final example, which also involves multiplication. You want to set up a screen in Mode 7 in which the first 20 lines have red ink on a cyan background and the next 4 lines have white on red. This can be done in a single X% loop which positions the cursor with VDU31,0,X% and continues with the new background (VDU134,157) and ink (VDU129). We can achieve the change after line 20 with a truth/falsity equation:

```
100 FOR X%=0 TO 23: VDU31,0,X%,134+(X%>19)*5,157,129-(X%>19)*6:NEXT
```

For the first 20 lines (X%=0 to 19), the bracketed equations yield 0, but thereafter their value is -1, thus changing the background colour to 129 (134+ -5) and the ink to 135 (129- -6).

As a final word, not all micros give truth the value of -1: some seem to prefer 1, which makes the

programmers life a little easier. If you find any other useful applications for truth and falsity, do write to Muesli and tell us. Or, put another way:

```
100 GOTO (110-(You find useful applications)*10)
110 Dont write to Muesli and tell us: GOTO 130
120 Write to Muesli and tell us
130 END
```

(Chris Jones)

Who CALLs the tune?

I am a teacher and writer in and around EFL. For three years I have used a computer in my work, for word-processing, note-taking, storing and copying things; I am fascinated by the machine itself - I play with it, learn new languages, read the trade press, use Prestel and bulletin boards - in short I am highly motivated towards it. The recent spate of books and articles about CALL should fill me with excitement. But

Computers are expensive and fragile. Administrators dont like teachers messing with them. Teachers dont want students messing with them. Because no school ever has enough equipment for everyone to use, pecking order determines who gets what. Typically, this is reflected in hardware hierarchy: the administrators get the IBMulator + hard disk, the maths and science departments have an RMZ or BBC (sometimes a whole local area network), Modern Languages have a couple of unexpanded BBCs or Spectrums if theyre lucky, and the remedial kids can play with the ZX-81 if the physics master hasnt turned it into a data-logger. In this way, the administration determines learning.

Software is expensive and time-consuming to write. Software skills are hard and time-consuming to acquire. Publishers seek to cut costs (which means cheap, easy and badly-written programs) and/or maximise sales (which means precious little on the education side) and/or maximise margins (which means hyper-costly software plus program protection that prevents you rewriting the sloppy bits). If software writers are not teachers, they have at best an imprecise notion of learning needs; if software writers are teachers, their output will be small. The possibility of students writing their own software (as many teachers encourage in the case of other media) is even smaller. In this way, materials are determined (even more than books and tapes) by publishers, advisors and the administration; seldom by classroom teachers, never by the learners themselves.

All but the most expensive setups have pitiable hardware resources: mass-storage of less than gigabyte proportions (and consequent processor capacity) makes for very limited human-machine interaction, highly deterministic programming, and lousy graphics (if language practice programs are at the stage of Lado, graphics output predates Lascaux). The machine determines what can be done by the programmer, the programmer determines what can be done with the program. Result: a range of preset branches. Anything more than this is very hard to achieve: try writing a draughts programs where the user can change the rules!

At the moment, computers are putting the methodological clock back. The more deterministic a program, the easier it is to write. The result? Drills, crude (very very crude) transformations, gap-fillers, clozes, one-to-one vocabulary correspondences, pre-categorising of items, lock-step learning paths (show me a program which takes account of the language I know before I start learning it), multiple choice questions (or their analogue, menus), text mazes.

If we had all we wanted in the way of cheap technology, programming expertise, and method-

ological humanity, we might, possibly, develop software, and the hardware to run it on, that encourages the learner to explore and develop language in any way s/he wants, e.g. by formulating and testing theories about the language, by teaching the machine, by searching for examples and ideas from an organizable database, by creating environments within the machine for student-centred games and simulations, by linking up with other learners round the world - at a cost that doesn't price 99% of potential users out.

Meanwhile (and I say this mainly because I still like computers, and so do many learners) let's put some effort into practical limited use of the machines we have, e.g. by writing simple word-processing utilities and fitting printers to our machines, so students can make/choose/change their traditional worksheets, handouts, transparencies etc; by writing within the learning group; by devising activities in which the computer plays a moderating or secretarial role (e.g. in simulations); by using networks to output different screens to a number of monitors in information-gap activities; by fitting modems and building up distance networks and viewdata hookups, and by developing cheap on-line information sources.

(John Morgan, Pilgrims, 8 Vernon Place, Canterbury)

CALL SOFTWARE

There is a constant demand for information about what software is available for EFL. In each issue from now on, we aim to devote a section to the principal pieces of software available in various categories: grammar, reading, testing, vocabulary work, adventures, word-processing etc. We would be grateful for any feedback from members concerning errors and oversights!

Recommended software is indicated with a *, a ** or a ***. Authoring packages are marked with (a), and software known to run on Econet with little or no modification with (n).

We kick off with a common CALL software type: activities involving reconstructing a corrupted, gapped or disordered text.

Text Reconstruction Programs

CLASS READER (Cambridge Language Arts Software): This is a flexible, and growing set of integrated software. Number 1 (Outcome) is a total text deletion program. Number 3 (Order, Order) is a jumbler. Both are very neatly programmed in Mode 7. Could be used as teletext pages. Texts can be written with Wordwise files (**, n, a (with CLASS WRITER)

CLOZEMASTER (Wida): Allows text up to 50 lines long. Deletion from every 5 to 15 words. Help and Cheat provided. Very popular and easy to use. (***, n, a)

COPYWRITE (ESM): Another Storyboard clone. (a)

GAPFIL (Pitmansoft): Creates gap-filling texts. (a)

GAPKIT (Camsoft): Creates gap-filling texts. Rather tatty in parts, and not too user-friendly, but very popular. (*, a, n)

MISSING LINKS (Sunburst): Clozed literary passages with 9 difficulty levels. (a)

PREDICTION (Camsoft): Student has to guess which word (of five) is likely to come next in a text. (a)

QUARTEXT (Longman): Four computer games, basically cloze and total cloze, but with excellent competitive and cheating elements. (**, a)

READAMATICS (Longman): A cloze package. If a word is not recognized, it is not rejected, but held for referral to the teacher. (n?)

SENTENCE SEQUENCING (Acornsoft): A range of rather iffy texts are scrambled by the micro, and have to be re-ordered. Data lines are easy to amend, but not authorable. (n)

STORYBOARD 2 (Wida): An upgrade of the ubiquitous (and invaluable) STORYBOARD. The help features have been expanded, to include not only prefixes and suffixes, but any letter of any word. It also allows text to be input from Wordwise or View. (***, a, n)

WORD SEQUENCING (Acornsoft): Re-ordering jumbled sentences. (n)

Next: Vocabulary Games and activities.

Buffer

Solutions to the Editor by December 6th.

The lucky winner will receive a copy of
Graham Davies new book!

TAILPIECES:

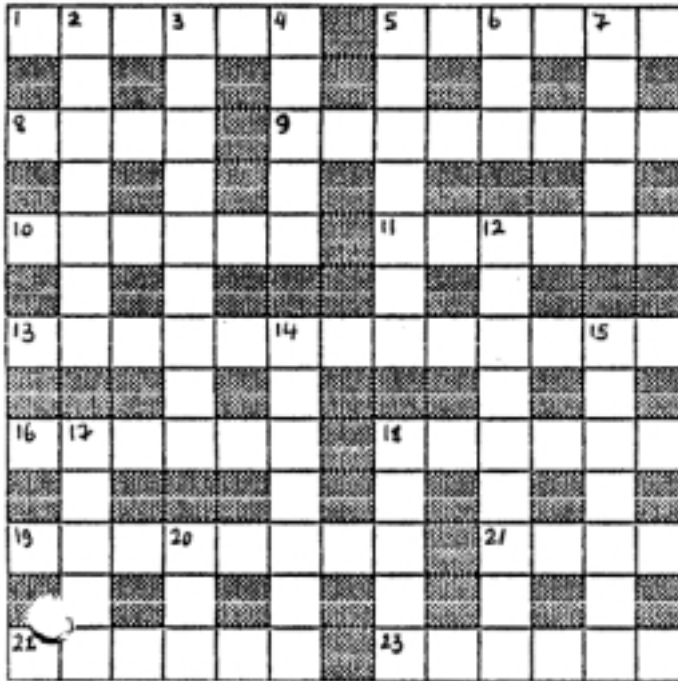
New Members/Renewals/Subscriptions: Please send a cheque for £10 to: Nick Reckert, MUESLI Membership Secretary, 5, Hills View, Great Shelford, Cambridge. (Tel: 0223-312393, work hours).

Contributions to the Newsletter, in the form of articles, reviews etc are most welcome. Contributions on disk in Wordwise format are ideal; otherwise A4 or half A4 in a legible typeface! Please send your copy to:
David Eastment, Bell College, South Road, Saffron Walden, Essex CB11 3DP.

Addresses/Contacts: Brian North is compiling a list of people working in CALL in EFL. If you think you should be on it, or know someone who should, please contact him at Davies School of English, 56 Eccleston Square, London SW1V 1PQ.

PAGE 1

PAGE 4



Across

1. Wasnt punished for jumping to line 255 (3,3)
5. How the programmer guards against eavesdroppers? (6)
8. Sounds funny, this particle (4)
9. Add-on handy for nomadic shepherds (5,3)
10. Dress up as a gap-filling ektherthithe? (6)
11. Number written in hexadecimal, ie in upper case (6)
13. Vulgar decision to make such crude drawings (3,10)
16. A1 tool no one repairs carelessly (6)
18. Could say why Evans thus described 22 (6)
19. Vetted by VDU (8)
21. At the centre of one sort of \$? (4)
22. Gadgets that take the squeals out of motorway crossings (6)
23. Does it use cereal printers? (6)

Down

2. Computer game has greeting up to start (7)
3. Horribly sour taste makes one avert gaze (9)
4. 22 think nothing of such statements (5)
5. Fingered, as 22 are (7)
6. 10 are all it can be (3)
7. Non-member solved Storyboard text aloud (5)
12. Put together in RAM with speed... (9)
14. ... for the enjoyment of musical CALL programmers? (7)
15. Preposition test undergoing evaluation (2,5)
17. Sounds like Arthurs sort of character (5)
18. It transmits Teletext (if M = 7)...(5)
20. ... on this, or to prevent crashing (3)