Vol.7 No.9 March 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

Multi-Precision Decimal Arithmetic

- COLOSSUS BRIDGE
- PAGE COMPOSITION
- ROLLER RALLY GAME
- 3D LANDSCAPES

BEEBUG Vol.7 No.9 March 1989

FEATURES

3D Landscapes 6
Page Composition for the BBC Micro 10
Dual Column Listings 18
Centres of Gravity (Part 2) 22
ADFS Wipe Utility 26
Multi-Precision Decimal Arithmetic 28
An Auto-Dating Utility 32
512 Forum 36
First Course - Ins and Outs of Basic (Part 3) 39
BEEBUG Education 44
Workshop - Game Strategy (Part 3) 46
File Handling for All (Part 9) 50
Roller Rally 55

REVIEWS

Heritage 15
Colossus 4 Bridge 34
ShareVAL - An Investment Analysis Package 42

REGULAR ITEMS

Editor's Jottings 4
News 5
Best of BEEBUG 58
Hints and Tips 59
Points Arising 61
Postbag 63
Personal Ads 64
Subscriptions & Back Issues 66
Magazine Disc/Cassette 67

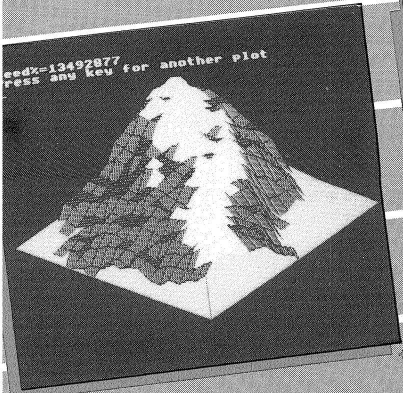
HINTS & TIPS

Closing Files Between Programs
Wordwrap
To Close Or Not To Close
Character Definitions
Modern Art

PROGRAM INFORMATION

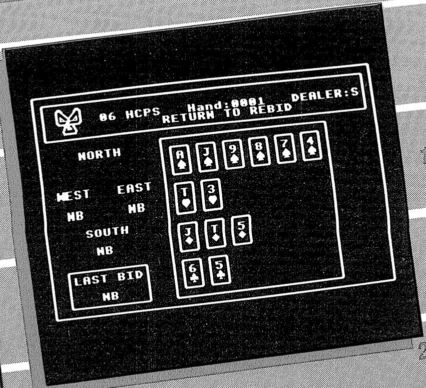
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



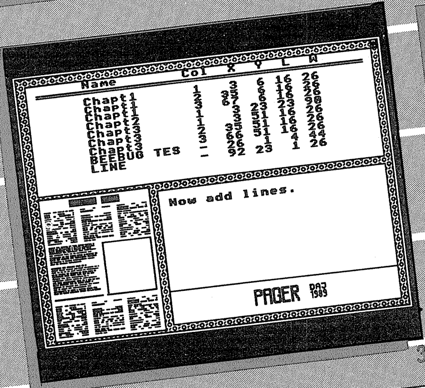
```
eed%=1349287?
ress any key for another plot
```

1.



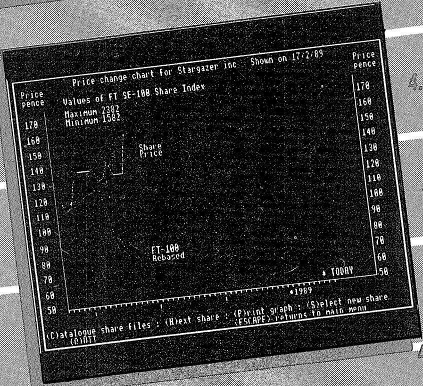
```
06 HCPS   Hand:0001      DEALER:S
        RETURN TO REBID
NORTH
WEST    EAST
NB      NB
  SOUTH
   NB
LAST BID
   NB
```

2.



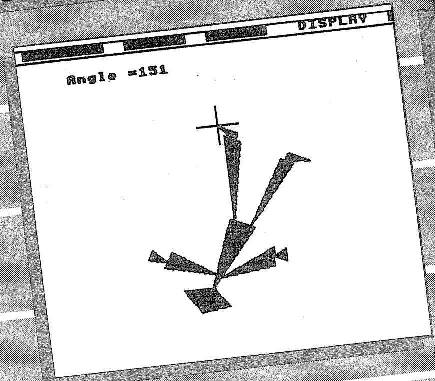
```
Name          Col   X   Y   L   W
Chapt1         1    3   6  16  26
Chapt1         2   35   6  16  26
Chapt1         3   67   6  16  26
Chapt2         1    3  23  23  90
Chapt3         1    3  51  16  26
Chapt3         2   35  51  16  26
Chapt3         3   66  51  16  26
BEEBUG TES     -   26   1   4  44
LINE           -   92  23   1  26

Now add lines.
```

3.

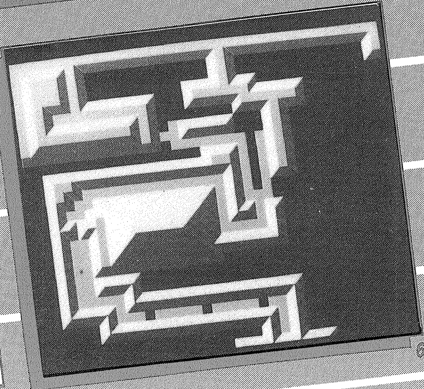


```
Price change chart for Stargazer inc   Shown on 17/2/89
Values of FT SE-100 Share Index
Maximum 2382
Minimum 1582
(C)atalogue share files : (N)ext share : (P)rint graph : (S)elect new share
(R)OTT                   (ESCAPE) returns to main menu
```

4.



5.



6.

1. 3D Landscape

2. Colossus Bridge

3. Page Composition

4. ShareVAL

5. Centres of Gravity

6. Roller Rally

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

SHOWS

Most BEEBUG members will be aware that last year, for the first time since it started, there was no Acorn User Show, previously the high point of the Acorn year. Now comes the news that Database Publications has decided to cancel the BBC Micro User Show planned for May.

Although Acorn's past support for specialist shows has often seemed somewhat grudging, the company has mounted a strong presence at shows such as the PC Show last September, and the recent Which Computer Show at the NEC Birmingham. Such shows are important for Acorn's continuing success, but can Acorn afford to go on ignoring many of its supporters as it has done in the past? Shows of all kinds are a vital barometer of the health of any market, and the absence of any specialist Acorn-based shows this year would only confirm the worst fears of Acorn's critics.

There are strong rumours that a new Acorn User Show will materialise this year, and Acorn's support and participation is surely vital. Let's hope that all concerned can work together to ensure a strong showcase for all whose business depends on the Acorn market.

WRITING TO BEEBUG

We always welcome your letters, comments and contributions, but it does help us if you follow a few simple rules. Make sure that your name, address, and membership number are clearly written. It also helps if you do not mix things like orders, queries etc. in the same letter (enclose two separate letters if necessary).

For contributions, a daytime telephone number can also be useful. If you send us a disc, do please enclose a covering letter, and tell us whether the disc is 40 or 80 track, DFS or ADFS. List all the files on the disc and their purpose, and tell us which word processor you have used (if this is appropriate). And please label any discs with your name and address too.

This may sound obvious, but it is surprising how many people fail to provide some or all of this necessary information, and we have even received discs with no accompanying information at all! Enclosing copies of any text files and listings of any programs is also much appreciated, and again helps us to evaluate contributions that much more quickly. Finally, we do ask all potential contributors to ensure that they follow the BEEBUG style as far as possible - a leaflet 'Notes of Guidance for Contributors' is available free of charge in response to an A5 SAE.

This month's telesoftware password is *strawberry*.

Looking ahead with BEEBUG

The following features are likely to be included in the April issue of BEEBUG:

Applications & Utilities:

- Mode 7 Histograms
- Disc Indexer
- Analysing RC and LR Circuits
- Decimal Arithmetic in Assembler
- Share Analysis
- File Handling for All
- Fast DFS Backup
- First Course - Large digit displays
- The Comms Spot
- Workshop
- 512 Forum

Reviews

- Music 5000 Junior
- Laser Printers

Plus News, Postbag, Hints and more.

RISC User

RISC User is the largest circulation magazine devoted entirely to the Acorn Archimedes range.
It is available on subscription to all BEEBUG members at a substantially reduced rate (see page 67 for details).

We expect the April issue to include:

Features:

- Ray Tracer
- Multi-task Notepad
- Procedure Libraries
- Function Keys
- RISC OS Sprite Calls
- Using the RISC OS Hourglass
- Archimedes Visuals
- Using ArcEdit

Reviews:

- Art Nouveau
- Archimedes Operating System Book
- Acorn's Unix Machine

and more.

RISC User is the ideal magazine to keep up to date with the Archimedes scene in every respect, and is particularly useful if you are contemplating the purchase of an Archimedes in the near future.

The latest details of the contents and distribution of both magazines are contained in the BEEBUG area of Micronet. Just type *BEEBUG# when on-line.

News News News News News News

SHOW SAGA

If you thought that the saga of last year's Acorn User Show went on and on, wait to see what happens this year. The current plans for shows this year go something like this:

> Database Exhibitions say that they will not hold any London shows at all.
>
> Redwood Publishing react by saying that there will be an Acorn User Show. It is rumoured that this will be at the London Business Centre - a nice venue, but you can't sell anything there, only demonstrate products. Redwood deny rumours and say that the Acorn User show will be held in June or July at Alexandra Palace - an up and coming venue for computer shows.
>
> Database decide to hold a London show in November after all.

It therefore looks likely that there will be two shows this year, one in June or July, and the other in November. However, I'm sure this isn't the end of the story, and BEEBUG will you keep you informed of goings on.

RISC OS LAUNCH DRAWS NEAR

Members who are waiting for the release of the new Archimedes operating system, RISC OS, before upgrading to an Archimedes, won't have much longer to wait. The original release date was set at some point in April. However, complete RISC OS packages consisting of the four ROM set, a completely new Welcome Guide and User guide, three Welcome discs, and an installation note have now been supplied to dealers for demonstration purposes. It is thought that Acorn will ensure adequate stocks are available before RISC OS is released to the public, although this could be as early as mid-March. All Archimedes supplied after the RISC OS launch will include it as standard, and anyone visiting BEEBUG's St. Albans' showroom can see a demonstration of the system.

MICRONET TELESOFTWARE

Regular users of Micronet, Prestel's micro-computing section, may have noticed a large reduction in the amount of telesoftware available for downloading. This has been brought about as a result of a decision by the Telemap Group, owners of Micronet, to withdraw chargeable telesoftware. Previously, if a charged program was downloaded, the cost of the program would be charged to the subscriber's account, and Prestel would pay that amount to Telemap, who would in turn give a percentage to the provider of the program. Telemap decided that the administrative costs of this were greater than the income from it, and therefore abandoned the system. Unfortunately, there is no other way for an information provider to charge for individual pages, and therefore suppliers who relied on the income from telesoftware have had to withdraw their support from Micronet. BEEBUG, however, will continue to provide free telesoftware for as long as possible.

4MATION ADDITIONS

Snatch, the screen grabber and printer dump from 4Mation, which was reviewed in BEEBUG Vol.7 No. 6 (page 22), has been upgraded to work with more printers. The package, which previously only supported Epson mono printers and the Integrex 132 Colourjet, will now drive all Epson and Star printers, including 24 pin types and colour printers, as well as all Epson compatibles. Also supported are the range of Canon colour ink-jet printers. The price of the Snatch remains unchanged at £23 inc. VAT. Existing users can obtain a free upgrade by returning their original disc to 4Mation Educational Resources, Linden Lea, Rock Park, Barnstaple, Devon EX32 9AQ, tel. (0271) 45566.

COMPUTERS ON THE BOX

Nearly two years after the demise of the Micro Live series, the BBC is planning a two and a half hour TV program for home computer users. The one-off, called Software Safari, is aimed at relative beginners, and will cover most models of micro, although Acorn machines will play a prominent role. It is rumoured that some programs from BEEBUG and RISC User might be featured. Software Safari will be transmitted on BBC2 on the morning of Sunday 2nd April.

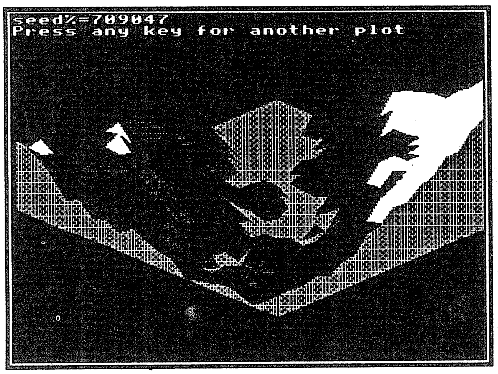
3D Landscapes

This program by John Greening exploits the graphics capabilities of the BBC micro to the full to display computer-generated landscapes.

The program presented here draws three dimensional landscapes. Each time you run the program you are likely to get a different picture because of the use of random numbers. Moreover, you will be able to influence the final outcome by controlling parameters in one or two of the procedures. The number of possible countrysides you can draw is very large indeed.

ENTERING THE PROGRAM

The program contains, among other things, three two-dimensional arrays that are very demanding of memory space. In consequence, it has been necessary to produce two versions of the program. The first (as listed here in its entirety) is for those who have added shadow RAM and the Graphics Extension ROM to their BBC model B, or who have a Master or Master Compact in which those facilities are built in.



```
seed%=709047
Press any key for another plot
```

3D landscape - Master version

The other version (for which modifications are given) is for those with a basic model B. However, everyone should type in the program 3DLands as listed, and save it away. This is then all that will be needed by those with the extra facilities mentioned above. Those of you with a model B should then make the alterations indicated in the listing at the end of the main program, and then save the revised version of the program. The changes are not so formidable as might appear at first sight. The revised versions of most lines are very similar to the original ones, and these will easily be edited with the Copy and cursor keys.

The omissions suffered by those with the basic model B are not great. You will not be able to put snow on the hills, and your "fields" will be a little larger with less colour variation. In compensation, the program will run faster. You will notice that this version of the program gets downloaded to &E00. For this to be successful you may have to kill any ROMS (other than the DFS) that claim workspace (an ideal application for the ROM Manager published in Vol.7 No.8).

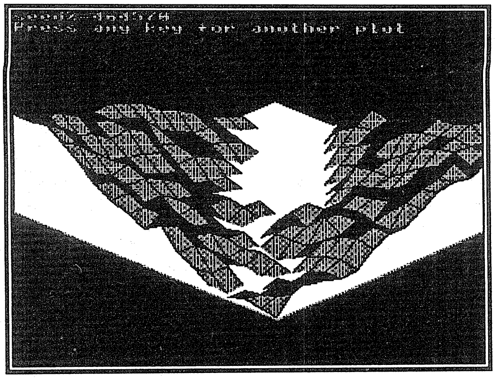
USING THE PROGRAM

If you run the program you will be told that the computer is calculating co-ordinates. This may take up to 20 seconds, so be patient. The computer then starts to draw. When it has finished you are likely to be looking down from a considerable height at an angle of about 45 degrees on a (partially) flooded valley with hills rising on either side to (snow-capped) peaks.

If you have run the main program your fields will be in four shades of green. The view is lit from the right front. The plain green fields are facing right, those with the lightest shading are facing front, and the darker fields have a leftwards inclination. The model B version has fields in yellow and green. The scene is to be regarded as being front lit with the yellow fields facing you and the green ones tending to slope away from you.

The areas with vertical green and black stripes are the faces of vertical sections through the ends of the terrain nearest you. You may get a better sense of realism by looking down on the screen at an angle instead of facing it directly.

At the top of the screen you will see a value for seed%. More of this later. You will be prompted to press a key for a re-run. Do this a few times in order to see the kind of variation you get in the final scene using the same set of parameters. The program does not take so long to re-run as it did when starting from the beginning, as it can use some of the previous calculations.



```
seed%=[illegible]
Press any key for another plot
```

3D landscape - model B version

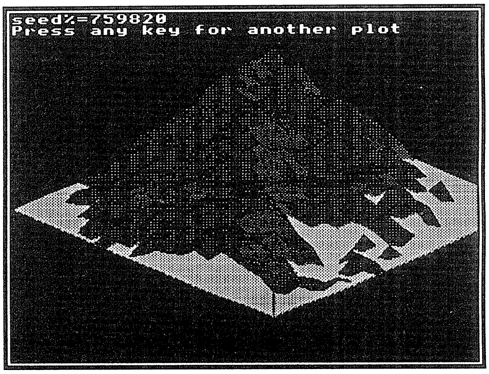
Now try some different parameters, e.g. 160, 30,and -15 in PROCverticals at line 240, and a value of 140 in PROCdrawmap (if a parameter is called for there) at line 270. You are now likely to find yourself looking down on a ridge with (snow-covered) peaks sloping down to water on either side. Run this a few times also to see the variations you can get.

Having got the idea, you can now start introducing your own parameter values. It is nearly always best to give opposite signs to the first and last parameters in PROCverticals, otherwise the terrain stays permanently underwater or rises steadily upwards with no sign of water, and there can be little doubt that water adds considerably to the attractiveness of the picture. You might also try 0,0,0 for the parameters. You could be flying over the great plains of North America cut up into their rectangular sections. If you set a small positive value for H% (say 50 or less) and an even smaller negative value for G% (say -3) the resulting pictures can be very different on different runs as the outcome is at the mercy of the random numbers combined with the roughness parameter R%.

As just indicated, the random numbers in the program mean that you do not get the same picture even if you use the same parameter values. However, if a picture particularly takes your fancy and you want to reproduce it, take a note of the value of seed% printed at the top of the screen. If you substitute this value for TIME in line 230, and run again with the same parameters, you will reproduce your picture. *The same 'random' numbers will have been* generated.

HOW IT WORKS

PROCnormalcoords sets up a chessboard of N%*N% squares. PROCviewingcoords twists the 'board' round through 45 degrees and tilts it away from you at 45 degrees. PROCverticals sets the value of H% as the height of the back corner of the 'board', and then uses this with random numbers, the 'roughness' parameter R%, and the gradient parameter G%, to run down both back edges of the 'board' calculating heights at every square. It then moves to inner rows of squares and does the same, but now using the heights of two neighbouring points as a reference. This stops unrelated variations in the two directions at right angles.



```
seed%=759820
Press any key for another plot
```

3D landscape - alternative view

PROCwaterlevel and PROCedgecolour try to take account of where blue water and vertical green and black stripes will be needed when the picture is finished. They succeed most of the time but occasionally fail. Colours are hard to come by in MODE 1. Hence the "poke" into memory byte &359 in the first line of PROCedgecolour to get the black and green stripes. PROCdrawmap plots the coloured squares. The REM statements in the program may be helpful in following its working.

PROCblankdeepwater cleans up the blue colour that can go outside the area of interest if the 'seabed' goes deep (it is the 'seabed' not the water surface that is plotted in blue).

Both versions of the program are provided separately on the monthly magazine disc/tape.

```
   10 REM Program 3Dlands
   20 REM Version B2.5
   30 REM Author  John Greening
   40 REM BEEBUG  March 1989
   50 REM Program Subject to copyright
   60 :
   70 REM H% is height of top central po
int above(+) or below(-) water level
   80 REM R% is a measure of the roughne
ss of the terrain
   90 REM G% is a measure of the gradien
t from centre to sides (+up, -down)
  100 REM S% is the height of the snowli
ne
  110 :
  120 MODE 129:VDU19,1,2,0,0,0
  130 ON ERROR GOTO 370
  140 VDU19,2,4,0,0,0
  150 PRINT TAB(8,16);"Calculating co-or
dinates"
  160 VDU29,640;450;
  170 X=450:Y=450:S=SQR(2)/2
  180 N%=20:REM Number of elements each
way
  190 DIM x(N%),y(N%),Z(N%,N%),a(N%,N%),
b(N%,N%)
  200 PROCnormalcoords
  210 PROCviewingcoords
  220 REPEAT
  230 seed%=TIME:dummy=RND(-seed%)
  240 PROCverticals(-50,30,20):REM H%,R%
,G%
  250 PROCwaterlevel
  260 PROCedgecolour
  270 PROCdrawmap(160):REM S%
  280 PROCblankdeepwater
  290 PROCdrawwaterline
  300 PRINT"seed%=";seed%
  310 PRINT"Press any key for another pl
ot"
  320 keyhit=GET
  330 PRINT'"Recalculating verticals"
  340 UNTIL FALSE
  350 END
  360 :
  370 MODE3:IF ERR<>17 REPORT:PRINT" at
line ";ERL
  380 END
  390 :
 1000 DEF PROCdrawwaterline
 1010 GCOL 0,1
 1020 IF Z(N%,0)>K% MOVE a(N%,0),Z(N%,0)
 ELSE MOVE a(N%,0),b(N%,0)/2
 1030 DRAW a(N%,0),b(N%,0)/2
 1040 DRAW a(N%,N%),b(N%,N%)/2:DRAW a(0,
N%),b(0,N%)/2:IF Z(0,N%)>b(0,N%)/2 DRAW
a(0,N%),Z(0,N%)
 1050 MOVE a(N%,N%),b(N%,N%)/2:IF Z(N%,N
%)>b(N%,N%)/2 DRAW a(N%,N%),Z(N%,N%)
 1060 ENDPROC
 1070 :
 1080 DEF PROCdrawmap(S%)
 1090 z=(b(0,0)-b(1,1))/2
 1100 FOR P=0 TO N%-1
 1110 FOR Q=0 TO N%-1
 1120 IF Z(P+1,Q+1)-Z(P,Q)+z>0 THEN c%=0
 ELSE c%=2
 1130 IF Z(P,Q+1)-Z(P+1,Q)+z>0 THEN d%=0
 ELSE d%=1
 1140 IF Z(P,Q)<b(P,Q)/2 THEN GCOL0,2 EL
SE IF Z(P,Q)>b(P,Q)/2+S% THEN GCOL0,3 EL
SE PROCshade(c%+d%):REM Blue if below wa
ter level and white if above snowline
 1150 MOVE a(P,Q+1),Z(P,Q+1)
 1160 MOVE a(P+1,Q+1),Z(P+1,Q+1)
 1170 PLOT85,a(P,Q),Z(P,Q)
 1180 IF Z(P,Q)>b(P,Q)/2+S% OR Z(P,Q)<b(
P,Q)/2 THEN GOTO 1230:REM Do not draw bl
ack lines round sectors if in snow or wa
ter
 1190 GCOL0,0
 1200 DRAW a(P,Q+1),Z(P,Q+1)
 1210 DRAW a(P+1,Q+1),Z(P+1,Q+1)
 1220 IF Z(P,Q)<b(P,Q)/2 THEN GCOL0,2 EL
SE IF Z(P,Q)>b(P,Q)/2+S% THEN GCOL0,3 EL
SE PROCshade(c%+d%)
 1230 MOVE a(P+1,Q),Z(P+1,Q)
 1240 MOVE a(P+1,Q+1),Z(P+1,Q+1)
 1250 PLOT85,a(P,Q),Z(P,Q)
 1260 IF Z(P,Q)>b(P,Q)/2+S% OR Z(P,Q)<b(
P,Q)/2 THEN GOTO 1300
 1270 GCOL0,0
 1280 DRAW a(P+1,Q),Z(P+1,Q)
 1290 DRAW a(P+1,Q+1),Z(P+1,Q+1)
 1300 NEXT:NEXT
 1310 ENDPROC
 1320 :
 1330 DEF PROCblankdeepwater
 1340 GCOL 0,0
 1350 MOVE a(N%,0),b(N%,0)/2
 1360 MOVE a(N%,0),2*b(N%,N%)/3:PLOT 85,
-a(N%,0)/3,2*b(N%,N%)/3
 1370 MOVE a(0,N%),b(0,N%)/2:
```

```
 1380 MOVE a(0,N%),2*b(N%,N%)/3:PLOT 85,
-a(0,N%)/3,2*b(N%,N%)/3
 1390 ENDPROC
 1400 :
 1410 DEF PROCnormalcoords
 1420 dX=2*X/N%:dY=2*Y/N%
 1430 FOR I=0 TO N%
 1440 x(I)=X-I*dX
 1450 y(I)=Y-I*dY
 1460 NEXT
 1470 ENDPROC
 1480 :
 1490 DEF PROCviewingcoords
 1500 FOR I=0 TO N%
 1510 FOR J=0 TO N%
 1520 a(I,J)=S*(x(I)-y(J))
 1530 b(I,J)=S*(x(I)+y(J))
 1540 NEXT:NEXT
 1550 ENDPROC
 1560 :
 1570 DEF PROCedgecolour
 1580 ?&359=&A
 1590 IF Z(0,N%)>b(0,N%)/2 THEN MOVE a(0
,N%),Z(0,N%) ELSE MOVE a(0,N%),b(0,N%)/2
 1600 MOVE a(0,N%),b(0,N%)/2
 1610 PLOT 85,a(0,N%-1),Z(0,N%-1)
 1620 PLOT 85,0,0
 1630 IF Z(N%,0)>b(N%,0)/2 THEN MOVE a(N
%,0),Z(N%,0) ELSE MOVE a(N%,0),b(N%,0)/2
 1640 MOVE a(N%,0),b(N%,0)/2
 1650 PLOT 85,a(N%-1,0),Z(N%-1,0)
 1660 PLOT 85,0,0
 1670 MOVE a(N%,0),b(N%,0)/2
 1680 MOVE a(0,N%),b(0,N%)/2
 1690 PLOT 85,a(N%,N%),b(N%,N%)/2
 1700 GCOL 0,2
 1710 IF Z(N%,0)<b(N%,0)/2 THEN MOVE a(N
%,0),b(N%,0)/2:MOVE a(N%,0),Z(N%,0):PLOT
 85,0,0
 1720 IF Z(0,N%)<b(0,N%)/2 THEN MOVE a(0
,N%),b(0,N%)/2:MOVE a(0,N%),Z(0,N%):PLOT
 85,0,0
 1730 ENDPROC
 1740 :
 1750 DEF PROCwaterlevel
 1760 CLS
 1770 GCOL 0,2
 1780 MOVE 0,b(0,0)/2:MOVE a(N%,0),0
 1790 PLOT 85,0,b(N%,N%)/2:PLOT 85,a(0,N
%),0
 1800 PLOT 85,0,b(0,0)/2
 1810 ENDPROC
 1820 :
 1830 DEF PROCverticals(H%,R%,G%)
 1840 Z(0,0)=H%
 1850 FOR I=1 TO N%
 1860 Z(I,0)=Z(I-1,0)+(RND(3)-2)*RND(R%)
+G%
 1870 NEXT
 1880 FOR J=1 TO N%
 1890 Z(0,J)=Z(0,J-1)+(RND(3)-2)*RND(R%)
+G%
 1900 NEXT
 1910 FOR I=1 TO N%
 1920 FOR J=1 TO N%
 1930 Z(I,J)=(Z(I-1,J)+Z(I,J-1))/2+(RND(
3)-2)*RND(R%)
 1940 NEXT:NEXT
 1950 FOR I=0 TO N%
 1960 FOR J=0 TO N%
 1970 Z(I,J)=Z(I,J)+b(I,J)/2
 1980 NEXT:NEXT
 1990 ENDPROC
 2000 :
 2010 DEF PROCshade(N%)
 2020 GCOL 16,0
 2030 IF N%=3 VDU23,12,1,1,1,1,1,1,1,1
 2040 IF N%=2 VDU23,12,1,1,0,1,1,1,0,1
 2050 IF N%=1 VDU23,12,1,0,1,0,0,1,0,1
 2060 IF N%=0 VDU23,12,1,0,0,0,0,1,0,0
 2070 ENDPROC
```

Those of you with a basic BBC model B should make the following changes to the listed program:

```
  105IF PAGE>&E00 GOTO 3000
  140VDU 19,3,4,0,0,0
  145COLOUR 2
  180N%=16:REM etc.
  270PROCdrawmap
 1010GCOL 0,2
 1080DEF PROCdrawmap
 1120IF Z(P+1,Q+1)-Z(P,Q)+z>0 THEN c%=1
ELSE c%=2
 Delete line 1130
 1140IF Z(P,Q)<b(P,Q)/2 THEN GCOL 0,3 EL
SE GCOL 0,c%
 1180IF Z(P,Q)<b(P,Q)/2 THEN GOTO 1230:R
EM Do not draw black lines round sectors
 if under water
 1220IF Z(P,Q)<b(P,Q)/2 THEN GCOL 0,3 EL
SE GCOL 0,c%
 1260IF Z(P,Q)<b(P,Q)/2 THEN GOTO 1300
 1700GCOL 0,3
 1770GCOL 0,3
 3000*TAPE
 3010FOR I%=0 TO TOP-PAGE STEP 4
 3020I%!&E00=I%!PAGE:NEXT
 3030?&13=?&13-(PAGE-&E00) DIV 256
 3040PAGE=&E00:RUN
```

Page Composition for the BBC Micro

David James presents the first part of a two part series on page composition and presentation using the BBC micro.

INTRODUCTION

During the course of this short series I shall be presenting a number of programs which together form a system for page composition and printing. This will allow any prepared text to be formatted and positioned on the page, using a variety of fonts and styles, for subsequent printing to produce a highly professional result akin to desktop publishing.

This month's article presents PAGER, comprising two main Basic programs (including some assembler for speed). This is a system which allows the user to build up a page of up to A4 size (i.e. 96 characters by 66 lines) on screen. It does this by formatting text files from any of the popular BBC word processors to a particular column width of your choice, and then positioning them on a map of the page. Headings may be added using a variety of heights and widths, and lines may be drawn in, horizontally and vertically, to make a border round the page or round separate parts of the page.

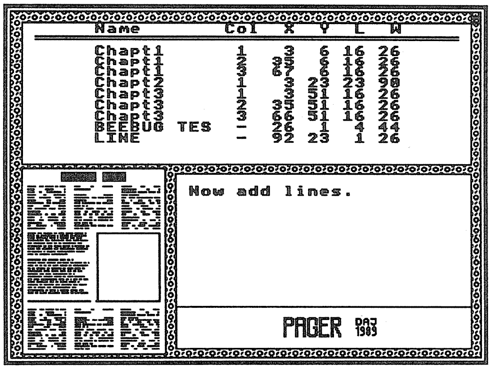
When the page design stage is complete, all relevant data is saved onto disc to be used by the printer driver (to be presented next month), which prints the page in several passes by first printing boxes and lines, then text, then headings. The standard PAGER set-up will run on an ordinary BBC model B with at least a 40 track disc drive and Epson-compatible printer, but a special printer driver will also be included for use with the PMS Multi-Font NTQ ROM, giving access to many different fonts and characters up to 256 times normal size.

This month we shall concentrate on the page design phase, and conclude the whole process next month with the font designer and printer driver.

THE PAGE COMPOSITION PROGRAMS

First of all, type listings one and two into your computer and save them immediately as PAGER1 and PAGER2 respectively. When PAGER1 is run, it sets up the mode 4 screen to be used by the main program, and then chains this in. First of all, you will be asked for a name to be assigned to this page. For certain tasks, only the first five letters of this name will be used, so be careful in order to avoid confusion later. You'll also be asked the name of the first text file to be formatted and positioned - make sure it exists on disc, and that it is terminated by a Return character (&0D in hex). Text files should be unformatted, and contain no embedded formatting or other commands. Most word processors such as View, InterWord and Wordwise Plus can do this.

Then type in the number of the font you want to use with this text, and the required height and width for the characters (a height and/or width of 1 give normal size characters - other values should be integers only representing multiples of the standard character height and width). If you are using the standard printer driver with only one font, the choice of font is irrelevant, but a response is still required (three zeros will do).



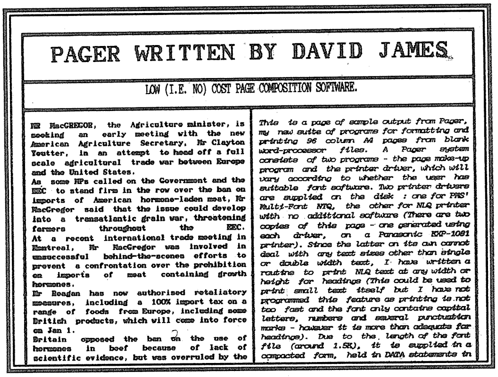
```
    Name           Col   X   Y   L   W
    Chapt1          1    3   6  16  26
    Chapt1          2   35   6  16  26
    Chapt1          3   67   6  16  26
    Chapt2          1    3  23  23  90
    Chapt3          1    3  51  16  26
    Chapt3          2   35  51  16  26
    Chapt3          3   66  51  16  26
    BEEBUG TES      -   26   1   4  44
    LINE            -   92  23   1  26

Now add lines.

PAGER DAJ 1989
```

Preparing an A4 page

Finally you will be asked for the column width (in characters) for the specified text. This must be at least 3 characters and cannot exceed 96 characters (the full page width). There will then be a pause as the computer loads the text file and formats it to the specified column width. When it finishes, you will be told the length of the resultant column - if this proves to be too long or too short for your purposes, you can re-load and re-format the text until you are happy with the result.

POSITIONING THE TEXT

At this point you can split the formatted text into any number of separate columns, and position these where you want on the map of the A4 page in the bottom left-hand corner of the screen. The cursor keys may be used to move the box within the page area, and Copy to fix it. As an aid, there is an audible bleep whenever the column box is central to the width of the page. Once a column has been fixed, there will be a short pause as the text is drawn out in miniature, and the process can be repeated with the remaining text from the original file. Once all the text has been positioned you can specify a further text file and continue with the same process. When you have positioned all the text you want, press Return at the *File name* prompt.


PAGER WRITTEN BY DAVID JAMES

LOW (I.E. NO) COST PAGE COMPOSITION SOFTWARE.

MR MacGREGOR, the Agriculture minister, is seeking an early meeting with the new American Agriculture Secretary, Mr Clayton Yeutter, in an attempt to head off a full scale agricultural trade war between Europe and the United States.
As some MPs called on the Government and the EEC to stand firm in the row over the ban on imports of American hormone-laden meat, Mr MacGregor said that the issue could develop into a transatlantic grain war, threatening farmers throughout the EEC.
At a recent international trade meeting in Montreal, Mr MacGregor was involved in unsuccessful behind-the-scenes efforts to prevent a confrontation over the prohibition on imports of meat containing growth hormones.
Mr Reagan has now authorised retaliatory measures, including a 100% import tax on a range of foods from Europe, including some British products, which will come into force on Jan 1.
Britain opposed the ban on the use of hormones in beef because of lack of scientific evidence, but was overruled by the

This is a page of sample output from Pager, my new suite of programs for formatting and printing 96 column A4 pages from blank word-processor files. A Pager system consists of two programs - the page make-up program and the printer driver, which will vary according to whether the user has suitable font software. Two printer drivers are supplied on the disk : one for PMS' Multi-Font NTQ, the other for NLQ printer with no additional software (There are two copies of this page - one generated using each driver, on a Panasonic KXP-1081 printer). Since the latter on its own cannot deal with any text sizes other than single or double width text, I have written a routine to print NLQ text at any width or height for headings (This could be used to print small text itself but I have not programmed this feature as printing is not too fast and the font only contains capital letters, numbers and several punctuation marks - however it is more than adequate for headings). Due to the length of the font file (around 1.5K), it is supplied in a compacted form, held in DATA statements in

Sample A4 page

Now type in any headings you want, followed by the font number (again, irrelevant if you only have one font, but you still need to type something), character height and width (important). A box representing the space required by that heading can be moved and positioned on the page as for text columns. Again, Return terminates this section of the program.

Finally, you will be given the chance to add any lines to the page. A dot will appear in the bottom left-hand corner of the page. The cursor keys can be used to move it around, and Shifted left and right cursor keys increase and decrease its length. Again, Copy fixes a line. When you have specified all the horizontal lines you want, press Escape and you will be able to add any vertical lines in the same way (except use Shifted up and down cursor keys to change length).

If you are constructing boxes, make sure that when you position the vertical sides of the box, the pixels at the corners of the box disappear (they will reappear when you press Copy) - otherwise the sides of the box will not meet at the corners. Escape ends the page composition process, and all relevant data is saved to disc for printing. The formatted columns are saved individually in directory P as the name of the page followed by a number, and the other page data is saved in directory D in a file named after the page. ADFS users will need to create the appropriate sub-directories in advance.

Note: before you start using PAGER, make sure you have enough raw text on disc, and a rough plan of where you want to put headings, articles, lines etc. The length of a column of text (lines of text) is roughly its size in bytes divided by the column width in characters, depending on the average length of a word. Unfortunately there is no hard and fast rule which will give a precise result.

MODIFYING THE PROGRAM

As listed, PAGER terminates when a single page has been completed. You could replace END at line 190 in PAGER2 by CH."PAGER1" to allow for additional pages to be composed in the same way. A procedure has also been included in PAGER2 (PROCchain) which, if called, will allow the printing of a page to take place immediately following page composition. We will deal with the use of that when the printer driver and font generator programs are described in next month's concluding article. For now I suggest you experiment with getting some page layouts ready for printing.

PAGE MUST be set to &1900 (PAGE=&1900) before loading and running these programs. This applies particularly to Master and Compact owners. PAGE is set to &1900 on the model B by default.

```
   10 REM Program PAGER1
   20 REM Version B1.02
   30 REM Author  David James
   40 REM BEEBUG  March 1989
   50 REM Program subject to copyright
   60 :
  100 MODE 4
  110 HIMEM=&3800
  120 VDU23;8202;0;0;0;
  130 VDU23,158,37,24,129,74,82,129,24,1
64
  140 VDU23,159,0,255,0,255,0,0,0,0
  150 VDU23,128,252,254,198,198,198,198,
254,252
  160 VDU23,129,124,254,198,198,198,198,
254,254
  170 VDU23,130,192,192,192,192,192,192,
192,192
  180 VDU23,131,198,198,198,198,198,198,
198,198
```

```
  190 VDU23,132,124,254,198,198,198,198,
198,192
  200 VDU23,133,254,254,192,192,192,192,
252,252
  210 VDU23,134,206,206,198,198,198,198,
254,124
  220 VDU23,135,192,192,192,192,192,192,
254,254
  230 VDU23,136,252,254,198,198,198,198,
254,252
  240 VDU23,137,240,216,216,204,204,198,
198,198
  250 VDU23,138,0,113,74,74,75,74,114,0
  260 VDU23,139,0,158,66,66,194,82,76,0
  270 VDU23,140,68,170,170,70,162,162,76
,0
  280 VDU23,141,36,106,42,38,34,34,44,0
  290 PROCdisplay
  300 CHAIN"PAGER2"
  310 :
 1000 DEF PROCbox(X1%,Y1%,X2%,Y2%)
 1010 x1%=32*X1%-4:x2%=32*X2%+32
 1020 y1%=((31-Y1%)*32)-4:y2%=((31-Y2%)*
32)+32
 1030 GCOL0,0:MOVEx1%,y1%:DRAWx1%,y2%
 1040 DRAWx2%,y2%:DRAWx2%,y1%
 1050 DRAWx1%,y1%:ENDPROC
 1060 :
 1070 DEF PROCwindow(X1%,Y1%,X2%,Y2%)
 1080 PROCbox(X1%,Y1%,X2%,Y2%)
 1090 VDU 28,X1%,Y1%,X2%,Y2%
 1100 COLOUR 129:COLOUR 0:CLS
 1110 ENDPROC
 1120 :
 1130 DEF PROCdisplay
 1140 COLOUR 129:COLOUR 0:CLS
 1150 FOR L%=0TO1239:VDU 158:NEXT
 1160 VDU26,11:PRINT STRING$(40,CHR$158)
 1170 GCOL0,0:MOVE28,12
 1180 DRAW416,12:DRAW416,560
 1190 DRAW28,560:DRAW28,12
 1200 GCOL0,1:MOVE32,16:MOVE412,16
 1210 PLOT85,32,556:PLOT85,412,556
 1220 PROCwindow(1,13,38,1)
 1230 PRINT"      Name        Col  X  Y
L  W"
 1240 PRINT" ";STRING$(36,CHR$159)
 1250 PROCwindow(14,30,38,15)
 1260 VDU31,9,13,128,129,132,133,136,32,
138,139
 1270 VDU31,9,14,130,131,134,135,137,32,
141,140
 1280 PROCwindow(14,26,38,15)
 1290 ENDPROC
```

```
   10 REM Program PAGER2
   20 REM Version B1.03
   30 REM Author  David James
   40 REM BEEBUG  March 1989
   50 REM Program subject to copyright
   60 :
  100 PROCcode:*FX14,6
  110 store%=&3800:dtable%=&1100
  120 htable%=&1200:horiz%=&1300
  130 vert%=&1400:*FX4,2
  140 PROCgetpagename:t%=0
  150 PROCgettext:PROCtitles
  160 PROClines:*FX13,6
  170 *FX15,1
  180 PROCsavedata:MODE7:*FX4
  190 END
  200 :
 1000 DEF PROCgetpagename
 1010 REPEAT PROCw2:CLS:*FX15,1
 1020 INPUT'" Page name  : " P$
 1030 UNTIL P$<>"":ENDPROC
 1040 :
 1050 DEF PROCgettext
 1060 t%=0:REPEAT
 1070 PROCw2:CLS:*FX15,1
 1080 INPUT'" File name  : " N$
 1090 IF N$="" GOTO 1380
 1095 IF NOT FNexists(N$) GOTO 1080
 1100 INPUT" Font Number: " TF%
 1110 INPUT" Height     : " TH%
 1120 INPUT" Width      : " TW%
 1130 REPEAT
 1140 CLS
 1150 REPEAT INPUT'" Column width : " wi
d%
 1160 UNTIL wid%>=3 AND (TW%*wid%)<=96
 1170 ch%=OPENINN$:L%=EXT#ch%:CLOSE#0
 1180 OSCLI"L."+N$+" "+STR$~store%
 1190 PRINT" Formatting-Please Wait."
 1200 lines%=FNformat(wid%)
 1210 PRINT" Article ";lines%;" lines lo
ng."
 1220 UNTIL FNok
 1230 done%=0:col%=1
 1240 REPEAT PROCw2:CLS:*FX15,1
 1250 PRINT'" ";lines%-done%;" lines to
be placed."
 1260 REPEAT INPUT " How many lines in t
his"'" column:-" this%
 1270 UNTIL this%>0 AND this%<=lines%-do
ne% AND (TH%*this%)<=68
 1280 PROCw1
 1290 PROCshowinfo(FNp(col%),TH%*this%,T
W%*wid%)
 1300 PROCmoveabox(TH%*this%,TW%*wid%)
 1310 IF esc% PRINTTAB(0,2+t%)SPC38:GOTO
1370
 1320 PROCstoredata
 1330 PROCsave
 1340 PROCroughidea(this%,adr%,TW%,TH%)
 1350 t%=t%+1:col%=col%+1
 1360 done%=done%+this%
 1370 UNTIL done%=lines% OR esc%
 1380 UNTIL N$="":?dtable%=t%-1
 1390 VDU7:ENDPROC
 1400 ENDPROC
 1410 :
 1420 DEF PROCsave
 1430 adr%=store%+(wid%+1)*done%
 1440 len%=(wid%+1)*this%
 1450 OSCLI"SAVE P."+LEFT$(P$,5)+STR$t%+
" "+STR$~adr%+" +"+STR$~len%
 1460 ENDPROC
 1470 :
 1480 DEF PROCsavedata
 1490 OSCLI"SAVE D."+LEFT$(P$,10)+" 1100
 1500"
 1500 ENDPROC
 1510 :
 1520 DEF FNformat(W%) len%=0
 1530 B%=&900:len%=0
 1540 adr%=store%
 1550 IF L%-1<=W% THEN $adr%=FNspread($a
dr%):=1
 1560 REPEAT
 1570 IF LEN$adr%=0 OR LEN$adr%>W% GOTO
1600
 1580 po%=1+LEN$adr%:PROCshift(W%-LEN$ad
r%)
 1590 GOTO 1650
 1600 IF adr%?W%=32 AND adr%?(W%-1)<>32
THEN po%=W%:GOTO 1640
 1610 po%=W%:REPEAT po%=po%-1:UNTIL adr%
?po%=32 OR po%=0
 1620 IF po%=0 po%=W%-1:PROCshift(2):adr
%?po%=ASC"-":adr%?W%=13:GOTO 1650
 1630 PROCshift(W%-po%)
 1640 adr%?po%=13
 1650 $adr%=FNspread($adr%)
 1660 adr%=adr%+W%+1
 1670 len%=len%+1
 1680 UNTIL adr%>=store%+L%
 1690 =len%
 1700 :
 1710 DEF FNspread($B%)
 1720 IF LEN($B%)=W% OR INSTR($B%," ")=0
 OR (LEN$B%)<W%/2 GOTO 1790
 1730 jump%=1
 1740 J%=1:REPEAT
 1750 IF J%?B%=32 THEN $(J%+B%)=" "+$(J%
+B%):J%=J%+jump%
 1760 J%=J%+1
 1770 UNTIL LEN($B%)=W% OR J%>=W%
 1780 IF J%>=W% AND LEN($B%)<W% THEN jum
p%=jump%+1:GOTO 1740
 1790 =$B%+STRING$(W%-LEN$B%," ")
 1800 :
 1810 DEF PROCcode
 1820 FOR I%=0 TO 2 STEP 2
 1830 P%=&A00
 1840 [OPT I%
 1850 .loop
 1860 LDY #0:LDA (&70),Y
 1870 LDY &74:STA (&70),Y
 1880 DEC &70:LDA&70:CMP#&FF:BNE L1
 1890 DEC &71:.L1
 1900 LDA &70:CMP &72:BNE loop
 1910 LDA &71:CMP &73:BNE loop
 1920 RTS
 1930 ]
 1940 NEXT
 1950 ENDPROC
 1960 :
 1970 DEF PROCshift(off%)
 1980 !&70=adr%+L%-1
 1990 !&72=adr%+po%-1
 2000 ?&74=off%
 2010 CALL&A00:L%=L%+off%
 2020 ENDPROC
 2030 :
 2040 DEF PROCoutline
 2050 MOVEX%,Y%:DRAWX%+W%,Y%
 2060 DRAWX%+W%,Y%+L%:DRAWX%,Y%+L%
 2070 DRAWX%,Y%:ENDPROC
 2080 :
 2090 DEF PROCroughidea(nl%,mem%,w%,h%)
 2100 GCOL0,0:VDU29,X%;Y%+L%;
 2110 FOR C%=0 TO nl%-1
 2120 line$=$mem%
 2130 mem%=mem%+1+LEN$mem%
 2140 FOR D%=0 TO -1+LENline$
 2150 IF MID$(line$,D%+1,1)=" " GOTO2210
 2160 PLOT69,4*D%*w%,-8*C%*h%
 2170 IF h%=1 AND w%=1 GOTO2210
 2180 PLOT0,4*w%-1,0
 2190 PLOT85,4*D%*w%,-8*C%*h%-8*(h%-1)
 2200 PLOT81,4*w%-1,0
 2210 NEXT,:VDU29,0;0;
 2220 ENDPROC
 2230 :
 2240 DEF PROCshowinfo(C$,sl%,sh%)
 2245 y%=FNyposn
 2250 PRINTTAB(6,y%);N$;
 2260 PRINTTAB(17,y%);C$;
 2270 PRINTTAB(27,y%);FNp(sl%);" ";FNp(s
h%);
 2280 ENDPROC
 2290 :
 2300 DEF PROCmoveabox(l%,w%)
 2310 L%=4+(l%-1)*8:W%=(w%-1)*4
 2320 xpos%=0:ypos%=68-l%:esc%=FALSE
 2330 X%=32:Y%=16:GCOL4,0:PROCoutline
 2340 PRINTTAB(21,y%);" 0 ";68-l%;
 2350 REPEAT
 2360 F%=4-12*INKEY-1
 2370 XI%=F%*((INKEY-26)-(INKEY-122))
 2380 YI%=F%*((INKEY-42)-(INKEY-58))
 2390 IF 0=(XI% OR YI%) GOTO 2510
 2400 PROCoutline
 2410 X%=X%+XI%:Y%=Y%+2*YI%
 2420 IF ABS(w%+(X%-32)/2-96)<=1 SOUND 1
,-15,200,1
 2430 IF X%<32 THEN X%=32
 2440 IF X%>412-W% THEN X%=412-W%
 2450 IF Y%<16 THEN Y%=16
 2460 IF Y%>=556-L% THEN Y%=556-L%
 2470 xpos%=(X%-32)/4
 2480 ypos%=68-1%-(Y%-16)/8
 2490 PRINTTAB(21,y%);SPC5;TAB(21,y%);FN
p(xpos%);TAB(24,y%);FNp(ypos%);
 2500 PROCoutline
 2510 IF INKEY-113 THEN esc%=TRUE
 2520 UNTIL INKEY-106 OR esc%
 2530 PROCoutline
 2540 ENDPROC
 2550 :
 2560 DEF PROCstoredata
 2570 adr%=1+dtable%+6*t%
 2580 adr%?0=TF%:adr%?1=TH%:adr%?2=TW%
 2590 adr%?3=xpos%:adr%?4=ypos%
 2600 adr%?5=this%
 2610 ENDPROC
 2620 :
 2630 DEF PROCtitles
 2640 head%=0:adr%=1+htable%
 2650 REPEAT:*FX15,1
 2660 PROCw2:CLS
 2670 PRINT'" Heading ";:INPUT LINE H$
 2680 IF H$="" GOTO 2810
 2690 INPUT" Font number:-" HF%
 2700 INPUT" Height      :-" HH%
 2710 INPUT" Width       :-" HW%
 2720 PROCw1:N$=LEFT$(H$,10)
 2730 PROCshowinfo(" -",HH%,HW%*LENH$)
 2740 PROCmoveabox(HH%,HW%*LENH$)
 2750 adr%?0=HF%:adr%?1=HH%:adr%?2=HW%
 2760 adr%?3=xpos%:adr%?4=ypos%
 2770 $(adr%+5)=H$
 2780 PROCroughidea(1,adr%+5,HW%,HH%)
 2790 adr%=adr%+6+LENH$
 2800 head%=head%+1:t%=t%+1
 2810 UNTILH$="":VDU 7
 2820 ?htable%=head%-1
 2830 ENDPROC
 2840 :
 2850 DEF PROClines
 2860 PROCw2:CLS
 2870 PRINT'" Now add lines."
 2880 PROCw1
 2890 PRINTTAB(6,2+t%)"LINE         -"
 2900 PROChorizlines
 2910 REPEAT UNTIL NOT INKEY-113
 2920 VDU7:PROCvertlines
 2930 REPEAT UNTIL NOT INKEY-113
 2940 VDU7:ENDPROC
```

```
 2950 :
 2960 DEF PROChorizlines
 2970 esc%=FALSE:X%=32:Y%=20:L%=4
 2980 mem%=horiz%+1:count%=0
 2990 REPEAT GCOL 4,0:PROCdrawh
 3000 REPEAT IF INKEY-113 esc%=TRUE
 3010 LI%=(INKEY-1)*((INKEY-122)-(INKEY-
26)):IF LI%=0 GOTO 3060
 3020 PROCdrawh:L%=L%+4*LI%
 3030 IF L%<4 L%=4
 3040 IF L%>416-X% L%=416-X%
 3050 GOTO 3160
 3060 IF INKEY-1 GOTO 3170
 3070 XI%=(INKEY-26)-(INKEY-122)
 3080 YI%=(INKEY-42)-(INKEY-58)
 3090 IF (XI% OR YI%)=0 GOTO 3170
 3100 PROCdrawh
 3110 X%=X%+4*XI%:Y%=Y%+8*YI%
 3120 IF X%<32 X%=32
 3130 IF X%>416-L%  X%=416-L%
 3140 IF Y%>556 Y%=556
 3150 IF Y%<20 Y%=20
 3160 PROCdrawh
 3170 UNTIL INKEY-106 OR esc%
 3180 IF esc% GOTO 3240
 3190 REPEAT UNTIL NOT INKEY-106
 3200 GCOL 0,0:PROCdrawh
 3210 xpos%=(X%-32)/4:len%=L%/4:ypos%=67
-(Y%-20)/8
 3220 PRINTTAB(21,2+t%);FNp(xpos%);" ";F
Np(ypos%);" ";FNp(len%)
 3230 PROClinedata(mem%):count%=count%+1
:mem%=mem%+3
 3240 UNTIL esc%:PROCdrawh
 3250 ?horiz%=count%-1
 3260 ENDPROC
 3270 :
 3280 DEF PROCvertlines
 3290 esc%=FALSE:X%=32:Y%=16:L%=8:mem%=v
ert%+1:count%=0
 3300 REPEAT:GCOL 4,0:PROCdrawv
 3310 REPEAT:IF INKEY-113 THEN esc%=TRUE
 3320 LI%=(INKEY-1)*((INKEY-58)-(INKEY-4
2)):IF LI%=0 GOTO 3370
 3330 PROCdrawv:L%=L%+8*LI%
 3340 IF L%<8 L%=8
 3350 IF L%>=560-Y% L%=560-Y%
 3360 GOTO 3470
 3370 IF INKEY-1 GOTO 3480
 3380 XI%=(INKEY-26)-(INKEY-122)
 3390 YI%=(INKEY-42)-(INKEY-58)
 3400 IF (XI% OR YI%)=0 GOTO 3480
 3410 PROCdrawv
 3420 X%=X%+4*XI%:Y%=Y%+8*YI%
 3430 IF X%<32 X%=32
 3440 IF X%>412 X%=412
 3450 IF Y%>=560-L% Y%=560-L%
```

Continued on page 61

Heritage

Reviewed by Chris Watts

Product	Heritage
Supplier	Bel Tech Ltd College House, St Leonards Close, Bridgenorth, Shropshire WV16 4EW. Tel. (0746) 765420
Price	£29.95 inc. VAT and p&p

INTRODUCTION

Researching family history is a fascinating and increasingly popular pastime. It generates a considerable amount of information, traditionally held on paper, which needs marshalling, analysing, revising, writing up and exchanging with fellow researchers. All are tasks which could be greatly assisted by a well-designed suite of programs.

Several programs are available for IBM PCs, but the BBC range is poorly served. Two programs were previously reviewed (BEEBUG Vol.7 No.1), including one (Bel Gen) from Bel Tech; so how does this current offering measure up?

The genealogist has three basic needs. The first is to hold lists of material searched and to be searched. The second is to hold abstracts of information found, to search this data and to link it to individual ancestors. The first need is probably best covered by a general database package; I know of only one package that caters for the second, and it runs on an IBM-PC. Heritage does not set out to address these two needs. The third need, which Heritage does support, is to hold a database of records about individual ancestors, which may be linked and relinked, as evidence becomes available.

THE PACKAGE

The package consists of an A5 ring-binder containing the 181 page manual and two 80 track discs. The manual is daisy-wheel printed, and the contents are quite adequate and well indexed. The default discs provided are configured for a model B with DFS. Instructions are given for configuring for other machines, for ADFS and for 40 track discs. Users unable to make the conversions, because of hardware limitations, should specify the required version when ordering otherwise they will need to return the discs and incur a small handling charge.

The copy available for review was version 1.6 - I understand that version 1.9 is now being supplied. However, despite a promise from the supplier, this version was not available for this review.

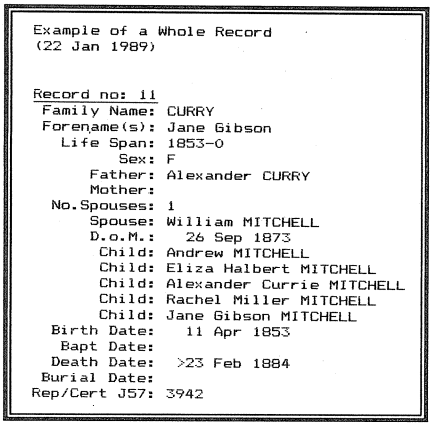
Accessing the sample file (ROYAL) may prove troublesome, so it is probably best to begin directly with creating your own.

GETTING STARTED

Having created a working copy for your configuration, the Setup utility is run. This asks what computer you have, whether you have one or two single/double sided or hard discs, and whether you wish to have the program disc permanently resident (recommended). Background and text colours, and default mode, may be specified, though model B users are restricted to mode 7, even with a shadow RAM board, which must be turned off - an infuriating restriction. Printer codes are also specified here; only parallel (Centronics) printers are supported; either condensed print or a wide carriage printer (or even better both) are desirable for this application.

The program can now be booted, and the main menu offers the options of Files, Data, Reports, Charts and Utilities. The next step is to create a record structure for holding your data (Create File). Thirty-two fields are available of which ten are preset: Family name, Forenames, Lifespan, Sex, Father, Mother, No. of Spouses, Spouse, Date of Marriage, and Children. Those for Spouse and Date of Marriage replicate themselves automatically as necessary to accommodate multiple wives. Similarly, more Children fields are automatically created - the limit is 23 wives plus children, but the number of wives may not exceed nine!

Additional fields may be specified; these may be Text, Date or Logical (i.e. empty or containing data). Date fields cater for approximate dates and before/after; quarters of the year (from St. Catherine's House indexes) are not accommodated. The Julian to Gregorian calendar change is catered for. Text and logical fields may be preset to be fixed at any length from 1 to 254, or left unlimited subject to the maximum record length (data actually entered) of 1000 characters (model B) or 2000 characters (other machines).



```
Example of a Whole Record
(22 Jan 1989)


Record no: 11
  Family Name: CURRY
  Forename(s): Jane Gibson
    Life Span: 1853-0
          Sex: F
       Father: Alexander CURRY
       Mother:
  No.Spouses: 1
       Spouse: William MITCHELL
       D.o.M.:   26 Sep 1873
        Child: Andrew MITCHELL
        Child: Eliza Halbert MITCHELL
        Child: Alexander Currie MITCHELL
        Child: Rachel Miller MITCHELL
        Child: Jane Gibson MITCHELL
   Birth Date:   11 Apr 1853
    Bapt Date:
   Death Date:  >23 Feb 1884
  Burial Date:
 Rep/Cert J57: 3942
```

Example of a whole record

Preset fields are fixed, and the replicating fields cannot be extended - this makes handling of data such as place of marriage, or divorce details, difficult. Fields cannot be deleted but only added, but their type or length can be changed, though changes to or from Date type do not convert existing data.

File size depends on computer and disc configuration. Typically a model B with data on both sides of an 80 track disc should hold about 850 records averaging 250 characters (memory limits the size of the file index held there); a Master (with more memory for indexing) could hold up to 1500 records using three disc surfaces.

ENTERING YOUR DATA

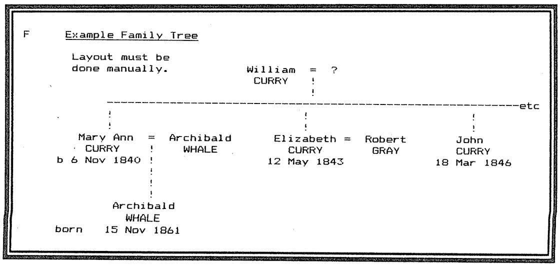
Data entry can now start, a task not to be underestimated; not a fault of the program however! The Edit screen presents your selected fields, and allows them to be filled in easily. Entries for Father, Mother, Spouse and Children may be made by typing in their names, which causes new records to be created automatically, or by selecting existing records from a pop-up index. Any links may be broken if fields are subsequently modified.

The Edit screen also provides the facility to browse through your file. Records may be examined sequentially, by selection from an index or via a Pedigree search. This latter allows you to step between parents, spouses and children, pausing to look at more detail via the Edit screen.

EXAMINING YOUR DATA

Having created a lineage-linked file containing details of ancestors and of others that may possibly need to be linked in later, the genealogist wishes to make searches and selective print-outs from it. Heritage provides a variety of facilities to do this.

The Reports sub-menu allows you to print the index, the contents of whole records, and details of family units. Tables may be printed, either of marriages (in preset format) or containing any desired fields; the specifications for the latter may be stored for future use. Records may be included on the basis of individual selection, by field content (AND/OR combinations allowed) or by logical search (i.e. does a logical field contain data or not). Index entries of records conforming to selection criteria may be listed to the screen using the Display List option. There is no facility to store or edit selection criteria.



```
F      Example Family Tree

        Layout must be
        done manually.          William  =  ?
                                 CURRY   !
                                         !
          -----------------------------------------------------------etc
          !                              !                       !
          !                              !                       !
    Mary Ann  =  Archibald       Elizabeth =  Robert          John
     CURRY    !    WHALE           CURRY      GRAY            CURRY
  b 6 Nov 1840 !                 12 May 1843               18 Mar 1846
              !
              !
          Archibald
            WHALE
  born    15 Nov 1861
```

Example of a family tree

The Report Writer provided is a simple screen editor, allowing records or fields to be intermingled with text. For most practical purposes this will need to be further edited using a word processor - this is possible.

The Construct Tree option, on the Charts sub-menu, allows classical family trees to be drawn, but not automatically. The user selects and positions each record, after specifying the required fields. Blocks of data may be deleted or moved; but I could not move them to the left. The adding of connecting lines and text is straightforward, and the results may be stored for future editing. This is in effect a simple word processor for family trees, and the results can be quite pleasing.

Birth Briefs (tree-like diagrams of parents, grandparents etc) are automatically generated with up to five generations. Diagrams tracing a pedigree, or lineage (backwards or forwards) may be produced using the pedigree search. Print-outs listing the five preceding generations or the descendants of any individual, giving lifespan, may be produced. The latter is particularly useful. All of these are in a predefined format.

```
Example of a Whole Record
(22 Jan 1989)


Record no: 11
  Family Name: CURRY
  Forename(s): Jane Gibson
     Life Span: 1853-0
           Sex: F
        Father: Alexander CURRY
        Mother:
   No.Spouses: 1
        Spouse: William MITCHELL
        D.o.M.:   26 Sep 1873
         Child: Andrew MITCHELL
         Child: Eliza Halbert MITCHELL
         Child: Alexander Currie MITCHELL
         Child: Rachel Miller MITCHELL
         Child: Jane Gibson MITCHELL
    Birth Date:   11 Apr 1853
     Bapt Date:
    Death Date:  >23 Feb 1884
   Burial Date:
 Rep/Cert J57: 3942
```

Example of a descendants' list

Another potentially useful output is the Record Card. This provides a pick-and-place capability for fields, together with added text. Selected records may then be printed in this format. Unfortunately, the version reviewed seems to contain a bug which made the placing of records and layout editing uncontrollable.

LIMITATIONS

No package of this nature can be expected to be perfect, so what are the practical limitations?

File size is not in reality a problem, as most genealogists compartmentalise their material in some way. Record size may be more of a limit on the model B. The inability to add to the automatically replicating fields is rather more serious.

Construct Tree is not automatic, as it is with one other package - but that is for the IBM PC. As a word processor for drawing family trees it obviously cannot rival the graphics specification of Pineapple's Diagram II, but then that is not linked to a lineage-linked database.

The real cruncher, however, is Heritage's inability to generate or accept data in GEDCOM format. This is the internationally accepted standard for the exchange of genealogical data between machines and packages. So once you have entered your data, there is no easy way to transfer it when you update. As genealogists keep data for literally decades, this is serious since re-entering the data will be error prone and could take many months.

No assistance phone service is available, queries must be sent to the suppliers in writing. Considering that the reviewer could not even get an up-to-date copy of the program, this does not bode well for the user. The most likely source of help is from fellow users, through the Society of Genealogists' Computer Users Group and its quarterly specialist magazine Computers in Genealogy.

CONCLUSIONS

Whilst Heritage is not the best genealogical software available, I know of nothing better, or likely to become available, for the BBC range. It does offer many useful features, and provided you can accept the lack of GEDCOM output, then it is worth purchasing.

The Society of Genealogists may be contacted at:
14 Charterhouse Buildings, London EC1M 7BA
or tel. 01-251 8799.

Dual Column Listings

This utility by Paul Pibworth will neatly and cleverly list your programs in two columns down the page, saving both time and money.

Have you ever thought how many trees have to be chopped down to make the listing paper which passes through your printer? And trees do take a long time to grow. If that has pricked your conscience, even just a little, then this utility will make you feel a little bit better. What really struck me was just how much unprinted paper there was on the right-hand side of a program listing. This was even more noticeable in the case of an assembler program. The answer, obvious I'm sure, is to use two columns for the listing.

To be properly readable, the right hand column on every page must continue from where the left column finishes. And if we are going to be at all clever about the way we try to achieve this (i.e. we don't just wind the paper back and print a second column alongside the first), then this means that the printer must print two completely different lines from the program every time the print-head scans across the sheet. In a typical printout, numbered according to the BEEBUG standard, line 10 and line 510 could be adjacent.

The utility given here does just this, and is written in assembler to achieve the necessary speed. It uses the Basic ROM to translate the keyword tokens into character strings, using a routine originally published by BEEBUG. The utility as listed will work correctly for Basic on the Master or Compact, but since there are differences between the various versions of BBC Basic, you will need to change line 250 for other machines as follows. For a BBC model B with Basic I use:

```
        rom1=&63:rom2=&80:rom3=&8A
```

or:

```
        rom1=&67:rom2=&80:rom3=&8A
```

for Basic II.

USING THE UTILITY

You first need to type in and save the listed program (with any modification needed for Basic I or II). When you run it, the routine is assembled, and a message displayed for you to copy to save the resulting code. This is given the name DLIST, but you could change the name if you wish.

Once assembled in this way, function key f0 will call the utility to print a dual column listing of the program currently in memory. At any other time you will first need to install the utility by entering:

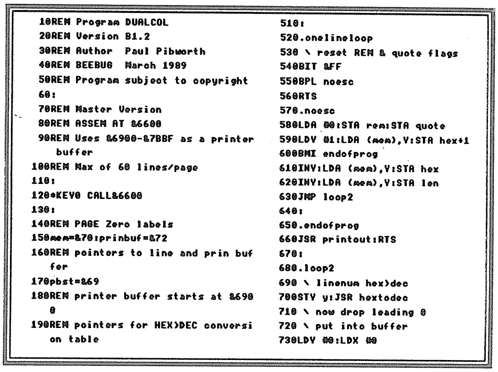
```
        *LOAD DLIST
```

and then:

```
        CALL&6600
```

each time you want to list out a program in memory. It is not possible to run the program from disc with *RUN DLIST because the program will then try to read the DFS ROM instead of the Basic ROM for the detokenising functions.

The program is not short, but if you use it, then your listings will look much neater and be physically shorter, thus saving money on paper, they will take less time to print (fewer carriage returns and less head movement), and you may even save a few trees. For those who are interested an explanation of the program follows.



```
   10REM Program DUALCOL                  510:
   20REM Version B1.2                     520.onelineloop
   30REM Author  Paul Pibworth            530 \ reset REM & quote flags
   40REM BEEBUG  March 1989               540BIT &FF
   50REM Program subject to copyright     550BPL noesc
   60:                                    560RTS
   70REM Master Version                   570.noesc
   80REM ASSEM AT &6600                   580LDA #0:STA rem:STA quote
   90REM Uses &6900-&7BBF as a printer    590LDY #1:LDA (mem),Y:STA hex+1
     buffer                               600BMI endofprog
  100REM Max of 60 lines/page             610INY:LDA (mem),Y:STA hex
  110:                                    620INY:LDA (mem),Y:STA len
  120*KEY0 CALL&6600                      630JMP loop2
  130:                                    640:
  140REM PAGE Zero labels                 650.endofprog
  150mem=&70:prinbuf=&72                  660JSR printout:RTS
  160REM pointers to line and prin buf    670:
    fer                                   680.loop2
  170pbst=&69                             690 \ linenum hex>dec
  180REM printer buffer starts at &690    700STY y:JSR hextodec
    0                                     710 \ now drop leading 0
  190REM pointers for HEX>DEC conversi    720 \ put into buffer
    on table                              730LDY #0:LDX #0
```

Sample dual column listing

UNDERSTANDING THE PROGRAM

The machine code is assembled at page &6600 to leave room for a printer buffer starting at page &6900. This buffer will be copied exactly to the printed page, so needs to be 60x80 bytes. There is also a one line buffer at page &A00.

The process used is as follows. A pointer moves through the program, starting from the first line. The line is detokenised and placed into &A00 (together with the line number). The detokenised line, together with its line number is now moved into the main buffer, but in such a manner that alternative 40 byte blocks are filled. Any partially filled blocks are treated as full, as each new line is moved across, i.e. a new block is begun. When 60 such blocks have been used, the inbetween blocks are then used (to form the second column), up to number 120. At this stage, the whole buffer is dumped to the printer, wiped clean, and the process begins again.

Although the main loop is based on one line at a time, a few matters need to be dealt with first. Thus, a title is placed in the first few bytes. This is not essential, but can be used to identify the assembled code. The maths pointers are set up for use in converting line numbers into decimal. The printer buffer is "cleaned", PAGE is read, and a flag (line) is set to keep a tally of how many 40 byte blocks have been used.

Each line must start with the setting of two flags that record the status of REM and quotes. Once a REM has been detected, program bytes are no longer detokenised. The same is true if the byte is enclosed within quotes. If the first byte (after &0D) is 255, this signals the end of the program, and the program exits, but not without printing the contents of the buffer. The next two bytes in a line represent the line number, and these are converted to decimal in a subroutine. This decimal number is first stored in a page zero location, before being placed in &A00 in a leading spaces, five digit format. The rest of the line is now read, byte by byte (.read), detokenised if necessary, (i.e. if a code is greater than 127), and placed into the short buffer.

Any references to line numbers within a program (GOTO etc.) are in a special tokenised format, and need further treatment. They need to be changed first into normal format line numbers, and then into decimal numbers, but without leading zeros.

When the end of a line is reached, the program moves to *.transfer*. The hardest part is over now. The contents of &A00 are moved into the next free block(s) in the printer buffer. If the buffer fills before this transfer is complete, then a branch is made to a routine to send the contents to the printer. The rest of the line is then transferred, the buffer pointers having been reset.

The print routine works through the buffer, converting blanks (CHR$0) to a space (CHR$32). The same treatment is given to any high byte codes (colour codes for example). The printer is turned on with the equivalent of VDU 2. As each byte is sent, it is replaced with a blank, thus saving a separate wipe at the end. At the end of a page, a 'form feed' is sent, before turning the printer off. This makes the printer run on to the next page, and is thus unsuitable as it stands for cut sheets. This could be changed by adding a JSR&FFE0 after line 2550, so that a key tap would be needed in order to continue.

The 40 byte blocks are actually arranged so that the last two bytes are blank, (to leave a gap between the columns). If a line runs over into a second block, then the first 5 bytes are kept blank, so that longer lines do not go under a line number. For example:

```
 1234REM this is a load of rubbish jus
      t to show what I mean.
```

A further refinement would be to include a space after a line number, eg.

```
 1234 REM etc etc
```

The program is well documented throughout so you should be able to relate the description above to the actual coding if you wish to do so.

```
   10 REM Program DUALCOL
   20 REM Version B1.2
   30 REM Author  Paul Pibworth
   40 REM BEEBUG  March 1989
   50 REM Program subject to copyright
   60 :
   70 REM Master Version
   80 REM ASSEM AT &6600
   90 REM Uses &6900-&7BBF as a printer
buffer
  100 REM Max of 60 lines/page
  110 :
  120 *KEY0 CALL&6600
  130 :
  140 REM PAGE Zero labels
  150 mem=&70:prinbuf=&72
  160 REM pointers to line and prin buff
er
  170 pbst=&69
  180 REM printer buffer starts at &6900
  190 REM pointers for HEX>DEC conversio
n table
  200 hex=&78:REM hold hex no.
  210 dec=&7A:REM hold dec no. &7A-&7E
  220 len=&7F:REM hold line length
  230 byte=&80:REM hold last character t
aken
  240 rem=&81:quote=&82:REM Flags for RE
Ms and Quotes
  250 rom1=&4C:rom2=&84:rom3=&81:REM BAS
IC 4 ROM token addresses
  260 address=&83:and=&85:REM hold addre
sses above
  270 y=&86:x=&87:REM hold X & Y
  280 line=&88:REM record of line being
printed (1-60)
  290 pagelen=60:REM no. of lines per pa
ge
  300 :
```

```
 310 C%=&6600
 320 FOR pass=0 TO 2 STEP 2
 330 P%=C%
 340 [OPT pass
 350 JMP start
 360  \ Title
 370 EQUS"BEEBUG DLIST UTILITY  "
 380 .start
 390 JSR pba \ print buff pointer
 400 LDX #pagelen:LDY #255:LDA #0
 410 .wipe2
 420 INY:STA (prinbuf),Y
 430 CPY #80:BNE wipe3:JSR inc80
 440 LDY #0:DEX:LDA #0
 450 .wipe3
 460 CPX #0:BNE wipe2
 470 JSR pba \ reset prinbuf pointer
 480 LDA #0 \ get page
 490 STA mem:LDA &18:STA mem+1
 500 LDA #0:STA line \ set line=0
 510 :
 520 .onelineloop
 530  \ reset REM & quote flags
 540 BIT &FF
 550 BPL noesc
 560 RTS
 570 .noesc
 580 LDA #0:STA rem:STA quote
 590 LDY #1:LDA (mem),Y:STA hex+1
 600 BMI endofprog
 610 INY:LDA (mem),Y:STA hex
 620 INY:LDA (mem),Y:STA len
 630 JMP loop2
 640 :
 650 .endofprog
 660 JSR printout:RTS
 670 :
 680 .loop2
 690  \ linenum hex>dec
 700 STY y:JSR hextodec
 710  \ now drop leading 0
 720  \ put into buffer
 730 LDY #0:LDX #0
 740 .loop3
 750 LDA dec,Y:CMP #48:BNE loop4
 760 LDA #32:STA &A00,X
 770 INX:INY:CPY #4:BNE loop3
 780 .loop4
 790 LDA dec,Y:STA &A00,X
 800 INX:INY:CPY #5:BNE loop4
 810 .loop5
 820 LDY y
 830  \ read and detokenise
 840  \ put into buffer (from&A00)
 850  \ X=position in &A00
 860  \ Y=position in orig line
 870 .read1
 880 LDX #5
 890 .read2
```

```
 900 INY:STY y \ store Y
 910 LDA (mem),Y \ read a byte
 920 STA byte \ store a byte
 930 CMP #13:BEQ get4 \ end of line?
 940 CMP #34:BNE iftoken \ 34=quote
 950  \ set/unset quote flag
 960 LDA #255:EOR quote:STA quote
 970 LDA byte \ reload the byte
 980 .iftoken \ is it a token?
 990 CLC:CMP #&80:BCS detok
1000 .buffer \ put into buffer
1010 STA &A00,X:INX
1020 .get3
1030 LDY y:JMP read2
1040 .get4
1050 JMP transfer
1060 :
1070 .detok
1080  \ check if REM flag set
1090 LDA #0:CMP rem:BEQ tok1
1100 LDA byte:JMP buffer
1110 .tok1
1120  \ check if quote flag set
1130 CMP quote:BEQ tok2
1140 LDA byte:JMP buffer
1150 .tok2
1160  \ check if a tokenised number
1170 LDA byte:CMP #&8D:BEQ numtoken
1180  \ set flag if a REM
1190 CMP #&F4:BNE tok4
1200 LDA #1:STA rem
1210 .tok4
1220  \ now look for token in ROM
1230  \ address holds TOKEN TABLE
1240  \ position, less 10
1250  \ Reset Basic Rom addresses
1260 LDA #rom1:STA address
1270 LDA #rom2:STA address+1
1280 LDA #rom3:STA and:LDY #10
1290 .tok5
1300 LDA (address),Y:CMP byte:BEQ table
1310 CLC:LDA address:ADC #1:STA address
1320 LDA address+1:ADC #0:STA address+1
1330 JMP tok5
1340 .table
1350  \ now go back to prev token
1360 DEY:LDA (address),Y
1370 CLC:CMP #&70:BCC table
1380  \ now go forward two, unless it
1390  \ is AND, ie preceded by &81
1400  \ (AND if preceded by &81)
1410 CLC:CMP and:BEQ table2:INY
1420 .table2
1430  \ read each byte
1440  \ put in buffer if < 127
1450 INY:LDA (address),Y
1460 CLC:CMP #&7F:BCS table3
1470 STA &A00,X:INX:JMP table2
1480 .table3
```

```
1490 JMP get3
1500  \ Detokenise a Tokenised number
1510  \ Store in location "hex"
1520  \ Convert from hex to dec
1530  \ Strip leading 0, put into &A00
1540 .numtoken
1550 INY:LDA (mem),Y
1560 ASL A:ASL A:PHA:AND #&C0
1570 INY:EOR (mem),Y:STA hex \ lowbyte
1580 INY:PLA:ASL A:ASL A
1590 EOR (mem),Y:STA hex+1 \ highbyte
1600 JSR hextodec:JSR dropzero
1610 JMP read2
1620 :
1630  \ TRANSFER TO PRIN BUF
1640 .transfer
1650 DEX:STX x:LDY #255:LDX #255
1660 .trans1
1670 INX:INY:CPY #38:BNE trans2
1680 JSR reset2
1690 .trans2
1700 LDA &A00,X:STA (prinbuf),Y
1710 CPX x:BNE trans1:JSR reset2
1720 CLC:LDA mem:ADC len:STA mem
1730 LDA mem+1:ADC #0:STA mem+1
1740 JMP onelineloop
1750 .reset2
1760 CLC:JSR inc80
1770 LDY #5:INC line:LDA #pagelen
1780 CMP line:BNE reset3
1790 LDA #40:STA prinbuf
1800 LDA #pbst:STA prinbuf+1
1810 .reset3
1820 CLC:LDA #pagelen*2
1830 CMP line:BNE reset4:JSR printout
1840 .reset4
1850 RTS
1860 :
1870 .printout
1880 JSR pba:TXA:PHA:TYA:PHA
1890 LDX #pagelen:LDY #0
1900 LDA #15:JSR &FFEE \ page off
1910 LDA #2:JSR &FFEE   \ VDU2
1920 .printout2
1930 BIT &FF
1940 BPL noesc2
1950 PLA:PLA:PLA:PLA
1960 JSR &FFE7:LDA #3:JMP &FFEE
1970 .noesc2
1980 LDA (prinbuf),Y:CMP #0:BNE char1
1990 LDA #32 \ convert 0 to space
2000 .char1
2010 CLC:CMP #127:BCC char2
2020 LDA #32 \ convert >127 to space
2030 .char2
2040 JSR &FFEE
2050  \ now replace with blank
2060 LDA #0:STA (prinbuf),Y
2070 INY:CPY #80:BNE printout2
2080 JSR inc80
2090 LDY #0:DEX:CPX #0:BNE printout2
2100 .printout3:LDA #13:JSR &FFEE
2110 LDA #12:JSR &FFEE
2120 LDA #3:JSR &FFEE
2130 PLA:TAY:PLA:TAX:JSR pba
2140 LDA #0:STA line \ reset line=0
2150 RTS
2160 :
2170  \ HEX>DEC SUBROUTINE
2180 .hextodec:LDA #48:STA dec
2190 STA dec+1:STA dec+2
2200 STA dec+3:STA dec+4
2210 TXA:PHA:LDY #255
2220 .htd2
2230 INY:CPY #5:BEQ htd6:LDX #48
2240 .htd3
2250 INX:LDA hex+1:CMP hibytes,Y
2260 BEQ htd4:BCS htd5:JMP htd2
2270 .htd4
2280 LDA hex:CMP lowbytes,Y
2290 BCS htd5:JMP htd2
2300 .htd5
2310 SEC:LDA hex:SBC lowbytes,Y:STA hex
2320 LDA hex+1:SBC hibytes,Y
2330 STA hex+1:STX dec,Y:JMP htd3
2340 .htd6:PLA:TAX:RTS
2350 :
2360 .dropzero
2370  \ Strip leading 0, put into A00
2380 LDY #255
2390 .htd7
2400 INY:CMP #4:BEQ htd8
2410 LDA dec,Y:CMP #48:BEQ htd7
2420 .htd8
2430 LDA dec,Y:STA &A00,X
2440 CPY #4:BEQ htd9:INX:INY:JMP htd8
2450 .htd9
2460 LDY y:INY:INY:INY:INX:RTS
2470 :
2480 .inc80:CLC
2490 LDA prinbuf:ADC #80:STA prinbuf
2500 LDA prinbuf+1:ADC #0:STA prinbuf+1
2510 RTS
2520 :
2530 .pba \ printer buffer address
2540 LDA #0:STA prinbuf
2550 LDA #pbst:STA prinbuf+1
2560 RTS
2570 :
2580 .lowbytes:EQUB16:EQUB232
2590 EQUB100:EQUB10:EQUB1
2600 .hibytes
2610 EQUB39:EQUB3:EQUB0:EQUB0:EQUB0
2620 ]
2630 NEXT
2640 PRINT"*SAVE DLIST "+STR$~C%+" "+ST
R$~P%
2650 END
```

Programs for Education

Centres of Gravity (Part 2)

We continue with our new feature on Programs for Education, by concluding an interesting examination of Centres of Gravity by David Lowndes Williams.

The Centres of Gravity package required a program that was too long to be written in one part. We presented the first part last month, and this allowed the creation of shapes to be manipulated with the second program. Also on last month's disc we supplied a shape file called "person"; this is again provided on the March disc/tape.

The program this month is called DISPLAY and is particularly interesting because it performs all of the centre of gravity calculations, and also displays the shape.

INDIVIDUAL CENTRES OF GRAVITY

Immediately upon entering the DISPLAY, program the centre of gravity of each triangle is calculated using PROCcalculate(c) as follows:

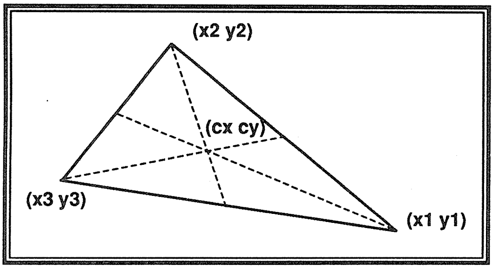


Figure 1

The centre of gravity of each triangle is 1/3 of the way up from the bisector of the base to the apex (see figure 1). The co-ordinates (cx,cy) of the centre of gravity are thus:

cx = (x2-x1)/3 + (x3-x1)/3

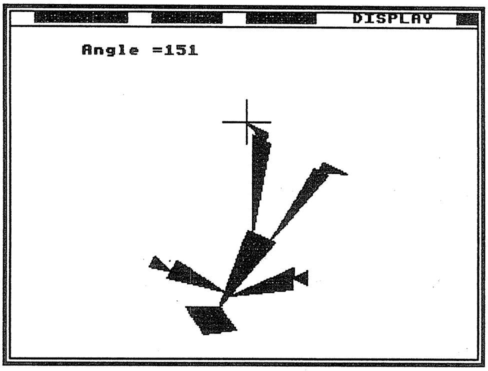
cy = (y2-y1)/3 + (y3-y1)/3

The calculations are performed in PROCcalculate(c), c being the number in the array of the triangle being calculated. This procedure also calculates the area of the triangle using Heron's formula:

A=SQR(s*(s-a)*(s-b)*(s-c))

where s=(a+b+c)/2

and a,b,c are the lengths of each side. The x co-ordinate for the centre of gravity of each triangle is stored in d%(6,c), the y co-ordinate in d%(7,c) and the area in d%(8,c) for future reference by the display routines.



A suspended figure

OVERALL CENTRE OF GRAVITY

The 'suspension' display requires further calculations as the shape has to be rotated by the correct angle to make the shape appear to be hanging correctly. PROCgeneral calculates the general centre of gravity for the whole shape, given the centres of gravity of the individual triangles. This is done by taking moments about the x and y axis (see figure 2).

Assume we are finding the overall centre of gravity for only two triangles, (for simplicity) and the first triangle has centre of gravity (x1,y1) and area A1, the second (x2,y2) and area A2. Also, we are assuming that weight is proportional to area and cancels from both sides of the equation. Therefore the area times the distance of the general centre of gravity from the y axis (xgen) equals the area*distance (x1*A1 for triangle 1) of each of the triangles. In other words, taking moments about the y axis gives:

(A1+A2)*xgen = x1*A1 + x2*A2

which rearranges to:

xgen =(x1*A1 + x2*A2)/(A1+A2)

Similarly we can replace the x values by y values to get the y co-ordinate (ygen) of the general centre of gravity. In this way the co-ordinates of the general centre of gravity are found, and the x coordinates stored in d%(6,0), the y coordinates in d%(7,0) and the area in d%(8,0).

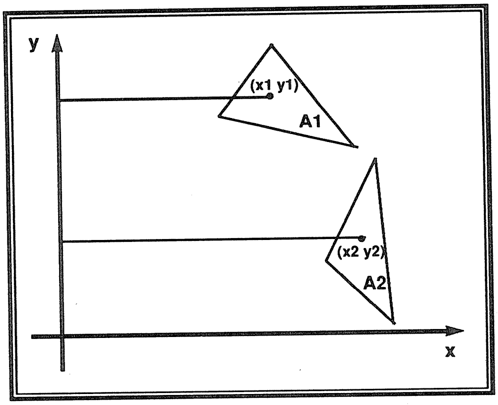


Figure 2

PROCdisp4 displays the rotated shape. The position of the centre of gravity is known and the point of suspension is requested. The angle between the line joining these points, and the vertical is calculated. The shape is then rotated by this angle to make it appear to hang. If the rotated centre of gravity is now above the point of suspension then 180 degrees are added to the angle to ensure the the shape hangs down, and doesn't hang up!

FNmenu(x,y,n) is used to perform the menu operations. It uses M$(n) to hold the titles of the items on the menu. All of these strings must be of the same length (in this case 12 characters long). The parameters x and y give the position of the text screen location of the top-left corner of the menu. This function could easily be used in other programs.

This program can easily be used without any understanding of how the maths and the programming work, so don't be put off if you don't follow either.

The screen dump procedure at line 2190 should be tailored to suit your needs. As it stands it works perfectly if you have the BEEBUGSOFT Dumpmaster ROM and are using an Epson-compatible printer. This dump could be stuck onto cardboard and cut out, so you can test the program to see if your shapes hang correctly. Happy hanging!

ERROR IN PART 1
Some readers may have found table 1 somewhat confusing. The y2's that appear in the column where C=1 should have been y1. We apologise for the error.

```
  10 REM Program DISPLAY
  20 REM Version B1.6
  30 REM Author  David Lowndes Williams
  40 REM BEEBUG  March 1989
  50 REM Program subject to copyright
  60 :
  70 ON ERROR GOTO 340
  80 maxt=INT((&1B00-PAGE)/45)
  90 DIM M$(7),d%(8,maxt)
 100 VDU24,0;0;1279;976;:*FX4,1
 110 VDU23,130,255,0,0,0,0,0,0,0,23,128
,1,1,1,1,1,1,1,1,23,129,128,128,128,128,
128,128,128,128
 120 PROCload_save("L","wkfile")
 130 IF d%(0,0)=0 E%=0:CHAIN"CENTRE"
 140 REPEAT
 150 PROCdisp2
 160 COLOUR128:COLOUR3
 170 PRINT TAB(28,0)" DISPLAY "
 180 M$(1)="Solid":M$(2)="Component"
 190 M$(3)="Centres":M$(4)="Suspension"
 200 M$(5)="Data":M$(6)="Screendump":M$
(7)="Out"
 210 b=FNmenu(27,3,7)
 220 IF b=1 PROCdisp1
```

```
  230 IF b=2 PROCdisp2
  240 IF b=3 PROCdisp3
  250 IF b=4 PROCdisp4
  260 IF b=5 PROCdisp5
  270 IF b=6 PROCdisp2:PROCscreendump
  280 IF b=7 E%=TRUE:PROCload_save("S","
wkfile"):CHAIN"CENTRE"
  290 COLOUR129:COLOUR3
  300 PRINTTAB(1,31)"Space to continue";
  310 G=GET:UNTIL FALSE
  320 END
  330 :
  340 IF ERR=17 AND INKEY-1 THEN 370
  350 IF ERR=17 GOTO 140
  360 REPORT:PRINT" at line ";ERL
  370 *FX4
  380 END
  390 :
  400 DEF FNmenu(x,y,do)
  410 LOCAL bk,fg,t,a,p,G:a=0
  420 bk=0:fg=3
  430 PRINT TAB(x,y-1)STRING$(12,CHR$95)
  440 FOR t=y TO y+do-1
  450 PRINT TAB(x-1,t)CHR$128;TAB(x+12,t
)CHR$129:NEXT
  460 PRINTTAB(x,y+do)STRING$(12,CHR$130
)
  470 FOR t=y TO y+do-1:a=a+1
  480 PRINT TAB(x,t)" "M$(a):NEXT
  490 *FX21,0
  500 *FX138,0,32
  510 p=0
  520 REPEAT:G=GET:COLOURfg:COLOURbk+128
  530 PRINT TAB(x,y+p-1)" "M$(p)
  540 IF G=139 p=p-1
  550 IF G=138 p=p+1
  560 IFp<1 p=1
  570 IFp>do p=do
  580 COLOURfg+128:COLOURbk
  590 PRINTTAB(x,y+p-1)" "M$(p)
  600 UNTILG=13:COLOURfg:COLOURbk+128
  610 =p
  620 :
  630 DEF PROCcursor(x,y,r):MOVEx-r,y
  640 DRAWx+r,y:MOVEx,y-r
  650 DRAWx,y+r:ENDPROC
  660 :
  670 DEF FNkeyboard:IF GET=13 =TRUE
  680 IFINKEY-58 y=y+4:IFINKEY-1 y=y+16
  690 IFINKEY-42 y=y-4:IFINKEY-1 y=y-16
  700 IFINKEY-26 x=x-4:IFINKEY-1 x=x-16
  710 IFINKEY-122 x=x+4:IFINKEY-1 x=x+16
  720 IF x<0 x=0
  730 IF x>1279 x=1279
  740 IF y<0 y=0
  750 IF y>980 y=980
  760 *FX21,0
  770 =FALSE
  780 :
  790 DEF PROCcalculate(c)
  800 LOCAL la,lb,lc,s,ax,ay,bx,by,cx,cy
,Area,vax,vay,vbx,vby,cntx,cnty
  810 ax=d%(0,c):ay=d%(1,c)
  820 bx=d%(2,c):by=d%(3,c)
  830 cx=d%(4,c):cy=d%(5,c)
  840 la=FNdist(bx,by,cx,cy)
  850 lb=FNdist(ax,ay,cx,cy)
  860 lc=FNdist(bx,by,ax,ay)
  870 s=(la+lb+lc)/2
  880 Area=SQR(s*(s-la)*(s-lb)*(s-lc))
  890 vax=(bx-ax)/3:vay=(by-ay)/3
  900 vbx=(cx-ax)/3:vby=(cy-ay)/3
  910 cntx=ax+vax+vbx:cnty=ay+vay+vby
  920 d%(6,c)=cntx:d%(7,c)=cnty
  930 d%(8,c)=Area
  940 PROCcursor(cntx,cnty,4)
  950 ENDPROC
  960 :
  970 DEF FNdist(x1,y1,x2,y2)=SQR((x1-x2
)^2+(y1-y2)^2)
  980 :
  990 DEF PROCdisp1:LOCALa
 1000 CLG:GCOL0,1
 1010 FOR a=1 TO d%(0,0)
 1020 MOVEd%(0,a),d%(1,a)
 1030 MOVEd%(2,a),d%(3,a)
 1040 PLOT85,d%(4,a),d%(5,a)
 1050 NEXT a:ENDPROC
 1060 :
 1070 DEF PROCdisp2:LOCALa
 1080 CLG:GCOL0,3
 1090 FOR a=1 TO d%(0,0)
 1100 MOVEd%(0,a),d%(1,a)
 1110 DRAWd%(2,a),d%(3,a)
 1120 DRAWd%(4,a),d%(5,a)
 1130 DRAWd%(0,a),d%(1,a)
 1140 NEXT a:ENDPROC
 1150 :
 1160 DEF PROCdisp3:LOCALa
 1170 CLG:GCOL0,3
 1180 FOR a=1 TO d%(0,0):GCOL0,1
 1190 MOVEd%(0,a),d%(1,a):MOVEd%(2,a),d%
(3,a):PLOT85,d%(4,a),d%(5,a)
 1200 GCOL0,3:MOVEd%(0,a),d%(1,a):DRAWd%
(2,a),d%(3,a):DRAWd%(4,a),d%(5,a)
 1210 DRAWd%(0,a),d%(1,a)
 1220 PROCcursor(d%(6,a),d%(7,a),4)
 1230 NEXT a
 1240 PROCgeneral
 1250 PROCcursor(d%(6,0),d%(7,0),128)
```

```
1260 ENDPROC
1270 :
1280 DEF PROCgeneral
1290 LOCAL a,Tarea,Tx,Ty
1300 Tarea=0:Tx=0:Ty=0
1310 FOR a=1 TO d%(0,0)
1320 Tarea=Tarea+d%(8,a)
1330 Tx=Tx+(d%(6,a)*d%(8,a))
1340 Ty=Ty+(d%(7,a)*d%(8,a))
1350 NEXT a
1360 d%(6,0)=(Tx/Tarea)
1370 d%(7,0)=(Ty/Tarea):d%(8,0)=Tarea
1380 ENDPROC
1390 :
1400 DEF PROCdisp4
1410 LOCALx,y,Ox,Oy,k,a
1420 CLG:PROCdisp3:COLOUR129:COLOUR3
1430 PRINT TAB(6,2)" Enter point of sus
pension "
1440 VDU19,2,8,0,0,0:GCOL3,2
1450 x=d%(4,0):y=d%(5,0)
1460 GCOL3,2:PROCcursor(x,y,64)
1470 REPEAT:Ox=x:Oy=y:k=FNkeyboard
1480 PROCcursor(Ox,Oy,64)
1490 PROCcursor(x,y,64)
1500 UNTIL k
1510 d%(4,0)=x:d%(5,0)=y
1520 CLG:GCOL0,3:PROCangle
1530 FOR a=1 TO d%(0,0)
1540 GCOL0,1:PROCplot(0,1,4)
1550 PROCplot(2,3,4):PROCplot(4,5,85)
1560 GCOL0,3:PROCplot(0,1,4)
1570 PROCplot(2,3,5):PROCplot(4,5,5)
1580 PROCplot(0,1,5)
1590 NEXT a
1600 PROCcursor(640,700,64)
1610 COLOUR128
1620 PRINTTAB(6,3)"Angle =";d%(1,0)
1630 ENDPROC
1640 :
1650 DEF PROCangle
1660 IF FNsmall(d%(4,0),d%(6,0)) AND FN
small(d%(5,0),d%(7,0)) PRINTTAB(10,2)" S
uspended from centre of mass "
1670 IFd%(4,0)=d%(6,0)d%(4,0)=d%(4,0)+1
1680 IFd%(5,0)=d%(7,0)d%(5,0)=d%(5,0)+1
1690 d%(1,0)=DEG(ATN((d%(4,0)-d%(6,0))/
(d%(5,0)-d%(7,0))))
1700 IF d%(7,0)>d%(5,0) d%(1,0)=d%(1,0)
+180
1710 ENDPROC
1720 :
1730 DEF PROCplot(t1,t2,m)
1740 LOCAL x,y,Ox,Oy
1750 x=d%(t1,a):y=d%(t2,a)
1760 REM rotate
1770 Ox=x-d%(4,0):Oy=y-d%(5,0)
1780 x=Ox*COS(RAD(d%(1,0)))-Oy*SIN(RAD(
d%(1,0)))
1790 y=Ox*SIN(RAD(d%(1,0)))+Oy*COS(RAD(
d%(1,0)))
1800 x=x*0.7:y=y*0.7:x=x+640:y=y+700
1810 PLOT m,x,y
1820 ENDPROC
1830 :
1840 DEF PROCload_save(d$,f$)
1850 LOCALf,a,b
1860 IF d$="S" f=OPENOUT f$ ELSE f=OPEN
IN f$
1870 FOR b=0 TO 5:IF b=1 THEN 1890
1880 IF d$="S" PRINT#f,d%(b,0) ELSE INP
UT#f,d%(b,0)
1890 NEXT
1900 FOR a=1 TO d%(0,0)
1910 FOR b=0 TO 5
1920 IF d$="S" PRINT#f,d%(b,a) ELSE INP
UT#f,d%(b,a)
1930 NEXT:NEXT
1940 CLOSE#f
1950 IF d$="S" OR d%(0,0)=0 ENDPROC
1960 PROCdisp2:FOR a=1 TO d%(0,0)
1970 PROCcalculate(a):NEXT
1980 PROCgeneral
1990 ENDPROC
2000 :
2010 DEF PROCdisp5
2020 LOCAL G$,a,b,@%
2030 CLG
2040 PRINTTAB(0,2)"To the printer (Y/N)
";:G$=CHR$(GET AND 223):PRINTG$
2050 VDU28,0,31,39,4
2060 IF G$="Y" VDU2
2070 @%=6
2080 PRINTSPC3"x1"SPC4"y1"SPC4"x2"SPC4"
y2"SPC4"x3"SPC4"y3"
2090 PRINT
2100 FOR a=1 TO d%(0,0):FOR b=0 TO 5
2110 PRINT d%(b,a);:NEXT:PRINT:NEXT
2120 PRINT'"Centre of mass: (";d%(6,0);
",";d%(7,0);")"
2130 PRINT"Area: ";d%(8,0)
2140 VDU3,10,10,28,0,31,39,0
2150 ENDPROC
2160 :
2170 DEF FNsmall(a,b)=ABS((a-b)/b)<0.05
2180 :
2190 DEF PROCscreendump
2200 *BPRINT
2210 ENDPROC
```

ADFS Wipe Utility

Does it take you forever to delete a number of files from an ADFS formatted disc? This program from Ian Kelly can help.

The Basic program presented here allows a selected group of files within a particular directory to be deleted in one go. You should start by typing in the listing and saving it. When run, the program assumes that the disc containing the files to be wiped is mounted. Therefore, if the Wipe program is on a different disc, the correct disc should be inserted after the program has been loaded, and *MOUNT entered.

On running the program, you are prompted for the pathname of the directory from which to delete the files. Either enter a directory name, or press Return to use the current directory. If you want to enter a star command, for example to catalogue the disc, press Space and then enter the command. A star at the start is not necessary.

The program then reads the names of all the files in the chosen directory, and for each file, asks whether it should be deleted or not. Pressing 'Y' will mark the file for deletion, though it will not actually be deleted at this stage. Pressing any other key skips over the file. To clarify the situation, ordinary files are shown in white, locked files in red and sub-directories in green. If a sub-directory is not empty, then this is shown, and that directory cannot be deleted. In the case of locked files and empty sub-directories, you are asked if you really want to delete that object.

Once you have replied Y or N to each file in the directory, the program prints a list of all the files that will be deleted, and asks for further confirmation. Pressing Y will wipe the files, while any other key will preserve them.

HOW IT WORKS

This program uses a call to the filing system routine OSGBPB with A=8 to read the names of all the objects in the chosen directory, and then uses OSFILE with A=5 to read the attributes for each object. These show whether the object is a file or a directory, and whether it is locked or not. The actual deletion is performed using the *DELETE command, preceded by *ACCESS if the object being deleted is locked.

```
   10 REM Program ADFS Wipe Utility
   20 REM Version B1.0
   30 REM Author  Ian Kelly
   40 REM BEEBUG  March 1989
   50 REM Program subject to copyright
   60 :
  100 MODE 7
  110 ON ERROR PROCerror
  120 fnames=&900:info=&B05
  130 wksp=&B34:osws=&B43:F%=0
  140 osgbpb=&FFD1:osfile=&FFDD
  150 FOR L%=&900 TO &B34 STEP4
  160 !L%=0:NEXT
  170 PROCdisplay
  180 VDU28,0,24,39,5,12
  190 PRINT" Which directory : ";
  200 INPUT LINE dir$
  210 IF LEFT$(dir$,1)=" " THEN OSCLI di
r$:PRINT:GOTO 190
  220 IF dir$<>"" THEN OSCLI("DIR "+dir$
)
  230 PROCgetcat
  240 IF F%=1 THEN PRINT'" There are no
files in this directory!"':GOTO 300
  250 PROCask
  260 IF F%=0 THEN PRINT''" No files!"':
GOTO 300
  270 PROCsure
  280 IF F%=0 THEN PRINT" Not deleted."'
:GOTO 300
  290 PROCdelete
  300 IF FNquest(" Another directory : "
)=1 THEN RUN
  310 PRINT
  320 END
  330 :
 1000 DEF PROCdisplay
 1010 FOR y=0 TO 1
 1020 PRINT TAB(y+1,y)CHR$157;CHR$132;CH
R$141;SPC(3-y);"ADFS Multiple file delet
er"
 1030 NEXT
```

```
1040 PRINT TAB(39,1)CHR$156 TAB(3,2)CHR
$157;CHR$130;" Green files are directori
es,     ";CHR$156 TAB(4,3)CHR$157;CHR$129
;"   Red files are locked.";SPC7;CHR$156
1050 VDU28,0,24,39,5,12
1060 ENDPROC
1070 :
1080 DEF PROCgetcat
1090 !(wksp+5)=47:!(wksp+9)=0:!(wksp+1)
=fnames
1100 X%=wksp MOD 256:Y%=wksp DIV 256
1110 A%=8:CALL osgbpb
1120 IF ?fnames=0 THEN F%=1:ENDPROC
1130 count=1:L%=fnames+1:REPEAT
1140 ?(L%+10)=13
1150 !wksp=L%:A%=5
1160 A%=USR(osfile) AND &FF
1170 ?(info+count)=(?(wksp+14) AND 8)
1180 ?osws=0
1190 IF A%=2 THEN ?(info+count)=?(info+
count)+2:OSCLI("DIR "+$L%):!(wksp+5)=1:!
(wksp+9)=0:!(wksp+1)=osws:A%=8:CALL osgb
pb:fe=?osws:OSCLI("BACK"):IF fe=&A THEN
?(info+count)=?(info+count)+4
1200 L%=L%+11:count=count+1
1210 UNTIL ?L%=0 OR L%>fnames+&205
1220 ENDPROC
1230 :
1240 DEF PROCask
1250 PRINT'" Do you want to delete the
following :"
1260 count=1:L%=fnames+1:F%=0:REPEAT
1270 PRINT
1280 PROCprintfn
1290 IF (?(info+count) AND 4)=4 THEN PR
INT;SPC9;CHR$131;"Not empty";:GOTO 1350
1300 IF FNquest(" : ")=0 THEN 1350
1310 IF (?(info+count) AND 2)=2 THEN IF
 FNquest("       Directory Y/N : ")=0 THE
N 1350
1320 IF (?(info+count) AND 8)=8 AND (?(
info+count) AND 2)<>2 THEN IF FNquest(CH
R$129+"      Locked Y/N : ")=0 THEN 1350
1330 ?(info+count)=?(info+count)+128
1340 F%=1
1350 L%=L%+11:count=count+1
1360 UNTIL ?L%=0 OR L%>fnames+&205
1370 ENDPROC
1380 :
1390 DEF PROCsure
1400 PRINT'" Delete :"'
1410 count=1:L%=fnames+1:Z%=0:REPEAT
1420 IF ?(info+count)<128 THEN 1460
1430 PRINT TAB(Z%);:PROCprintfn
1440 Z%=Z%+13
1450 IF Z%=39 THEN Z%=0:PRINT
1460 L%=L%+11:count=count+1
1470 UNTIL ?L%=0 OR L%>fnames+&205
1480 PRINT''" Are you sure : ";
1490 SOUND1,-10,150,1
1500 TIME=0:REPEAT UNTIL TIME>75
1510 *FX21,0
1520 F%=FNquest("")
1530 PRINT'
1540 ENDPROC
1550 :
1560 DEF PROCdelete
1570 PRINT" Deleting :"'
1580 count=1:L%=fnames+1:Z%=0:REPEAT
1590 IF ?(info+count)<128 THEN 1650
1600 PRINTTAB(Z%);:PROCprintfn
1610 IF (?(info+count) AND 8)=8 THEN OS
CLI("ACCESS "+$L%)
1620 OSCLI("DELETE "+$L%)
1630 Z%=Z%+13
1640 IF Z%=39 THEN Z%=0:PRINT
1650 L%=L%+11:count=count+1
1660 UNTIL ?L%=0 OR L%>fnames+&205
1670 PRINT'
1680 ENDPROC
1690 :
1700 DEF PROCprintfn
1710 col=135
1720 IF (?(info+count) AND 8)=8 THEN co
l=129
1730 IF (?(info+count) AND 2)=2 THEN co
l=130
1740 PRINT;CHR$col+$L%;
1750 ENDPROC
1760 :
1770 DEF FNquest(mess$)
1780 PRINT mess$;
1790 Z%=GET:IF Z%=89 OR Z%=121 THEN PRI
NT;"Y";:=1
1800 PRINT;"N";:=0
1810 :
1820 DEF PROCerror
1830 IF ERR=214 OR ERR=204 THEN PRINT'"
 No such directory"':IF FNquest(" Anothe
r directory : ")=1 THEN RUN
1840 IFERR=17 PRINT:IF FNquest(" Anothe
r directory : ")=1 THEN RUN
1850 PRINT
1860 IF ERR<>214 AND ERR<>17 THEN REPOR
T:PRINT;" at line ";ERL
1870 END
```

Multi-Precision Decimal Arithmetic (1)

A set of machine code routines to perform floating point arithmetic from Richard Beck.

While the floating point arithmetic provided by Basic is adequate for most purposes, there are times when it is too limiting. In particular, the accuracy is restricted to nine or ten significant figures, and the range is also fairly restricted. Additionally, if you want to use floating point arithmetic in machine code programs, you must either write your own routines, or call those in Basic. The first method is long-winded, while the second is clumsy and prone to compatibility problems.

The machine code routines given here and in next month's concluding article will perform the four rules of arithmetic (addition, subtraction, multiplication and division) on floating point numbers with up to 128 significant figures. These routines can easily be incorporated in your own programs, be they Basic or machine code, and a simple Basic program is given to demonstrate their use.

ENTERING THE PROGRAM

There are two listings given here. The first generates the actual machine code routines, while the second is a demonstration program that allows the results to be compared with those produced by Basic. Because of the length of the machine code routines, only those for addition and subtraction are included this month. The necessary additions for multiplication and division will be given next month.

Listing 1 should be typed in first, and saved. Note the jump in line numbers at two places (after lines 380 and 1960). This is to allow for next month's additions. Running this program will assemble the arithmetic routines and save the machine code under the name 'FPcode'.

Listing 2 is the demonstration program to illustrate the use of the new arithmetic routines. This should also be typed in and saved before running.

USING THE PROGRAMS

To try out the multi-precision routines, run the program of listing 2. This will first of all try to load the machine code produced by listing 1. Therefore, make sure that you run the first program to assemble the code, before running the demonstration.

You will first of all be prompted for the number of significant figures that you want the program to operate to. This doesn't alter the range of possible numbers, just the accuracy to which numbers are manipulated. The value can be anything from 4 to 128, although because of the way the program works, it must be an even number. In fact, if an odd number is entered, it will be rounded up to the next even number anyway.

Next, the program prompts for the first operand. All numbers are entered and displayed in standard form or (scientific notation), for example $1.234*10^7$. Instead of entering this in the notation that Basic uses, which would be 1.234E7, the part before the E (the mantissa), and the power of ten (the exponent) are entered as two separate numbers, pressing Return after each. Furthermore, the decimal point of the mantissa is not entered. Instead, it is assumed to be positioned between the first and second digits (if it wasn't, the number would not be in standard form). Therefore, the value $1.234*10^7$ would be entered into this program as two separate numbers, 1234 and 7. The number of digits in the mantissa should be no more than the number of significant figures specified at the start. If it is less, the program pads the value with zeros. The allowable range for the exponent is -126 to 126.

After the first operand, the operator should be entered. This is one of '+', '-', '*' or '/', followed by Return. Any other character (or indeed '*' and '/' until the routines are added) will cause an error when the equation is evaluated. Finally, the second operand is entered in the same way as the first. The program then calls the machine code routine to perform the operation, and prints the result. For comparison, the result produced using Basic's own floating point operations is also printed. Pressing any key will allow another sum to be performed, using the same precision. To change the precision, leave the program with Escape and and re-run it.

Next month we will cover the machine code needed for multiplication and division. We will also explain how these multi-precision routines can be incorporated in your own programs, as well as explaining in detail the principles of floating-point arithmetic, and the workings of these routines.

Listing 1

```
 10 REM Program FP Arithmetic
 20 REM Version B1.0
 30 REM Author  R.A. Beck
 40 REM BEEBUG  March 1989
 50 REM Program subject to copyright
 60 :
100 DIM code 2000
110 FOR pass=0 TO 3 STEP 3
120 P%=&6000
130 O%=code
140 [OPT pass + 4
150 JMP start
160 .wlen EQUB 0
170 .operator EQUB 0
180 .first EQUS STRING$(66,CHR$0)
190 .second EQUS STRING$(66,CHR$0)
200 .result EQUS STRING$(128,CHR$0)
210 .result_power EQUB 0
220 .power_diff:.offset EQUB 0
230 .shift_flag EQUB 0
240 .remain:.p_prod_over EQUB 0
250 .p_prod EQUS STRING$(64,CHR$0)
260 .sign EQUB 0:.nibble EQUB 0
270 .second_index EQUB 0
280 .pres_digit:.result_index EQUB 0
290 .wlenml EQUB 0
300 .wlenpl EQUB 0
310 .wlent2 EQUB 0
320 .start LDX wlen:INX
330 STX wlenpl:DEX:DEX
340 STX wlenml:LDA wlen:ASL A
350 STA wlent2:TAX:DEX:LDA #0
360 STA result_index
370 .zero_loop STA result,X:DEX
380 BPL zero_loop:LDA operator
430 .next_op2:CMP #ASC("+")
440 BNE next_op3:JMP add_check
450 .next_op3:CMP #ASC("-")
460 BNE next_op4:JMP sub_check
470 .next_op4:BRK:BRK
480 EQUB &A
490 EQUS ("Invalid operator")
500 EQUD &A0D:EQUB 0
510 .add_check LDA first+65
520 BNE ac_fneg:LDA second+65
530 BNE ac_fpos_sneg:JMP addition
540 .ac_fneg LDA second+65
550 BEQ ac_fneg_spos:JMP addition
560 .ac_fneg_spos JMP subtraction
570 .ac_fpos_sneg JMP subtraction
580 .sub_check LDA first+65
590 BNE sc_fneg:LDA second+65
600 BEQ sc_fpos_spos:LDA #0
610 STA second+65:JMP addition
620 .sc_fneg LDA second+65
630 BNE sc_fneg_sneg:LDA #&FF
640 STA second+65:JMP addition
650 .sc_fneg_sneg JMP subtraction
660 .sc_fpos_spos JMP subtraction
670 :
680 .addition JSR add_sub_init:SED
690 LDY wlenml:CLC
700 .add_loop LDA first,Y
710 ADC result,Y:STA result,Y:DEY
720 BPL add_loop:BCC no_shift
730 LDA result_power:CMP #&FF
740 BNE not_max:JMP max_power
750 .not_max JSR shift_result_right
760 LDA result:ORA #&10:STA result
770 INC result_power
780 .no_shift LDA first+65:STA sign
790 JMP left_justify
800 .shift_result_right LDY wlen
810 JMP shift_entry
820 .shift_loop LDA result,Y:ASL A
830 ASL A:ASL A:ASL A:INY
840 ORA result,Y:STA result,Y:DEY
850 .shift_entry:LDA result,Y
860 LSR A:LSR A:LSR A:LSR A
870 STA result,Y:DEY:BPL shift_loop
880 RTS
890 .subtraction JSR add_sub_init
900 SED:LDA power_diff
910 BNE sub_no_swap:SEC:LDA #0
920 LDY wlen:SBC result,Y:DEY
930 .sub_check_loop LDA first,Y
940 SBC result,Y:DEY
950 BPL sub_check_loop:BCS sub_no_swap
960 JSR swap_first_second
970 JSR add_sub_init
980 .sub_no_swap SEC:LDA #0:LDY wlen
```

```
 990 SBC result,Y:STA result,Y:DEY
1000 .sub_loop LDA first,Y:SBC result,Y
1010 STA result,Y:DEY:BPL sub_loop
1020 BCC sub_error:LDA first+65
1030 STA sign:JMP left_justify
1040 .sub_error:BRK:BRK
1050 EQUS "Subtraction error":EQUB &A
1060 EQUB 0
1070 :
1080 .left_justify CLD:LDA result
1090 AND #&F0:BEQ justify:JMP round_off
1100 .justify LDX #0
1110 .left_loop1:LDA result,X
1120 BNE move_left:INX:CPX wlen
1130 BNE left_loop1:LDA #&80
1140 STA result+64:LDA #0:STA result+65
1150 RTS
1160 .move_left CPX #0:BEQ MSD_filled
1170 STX offset:ASL A:STA shift_flag
1180 LDA result_power:SEC
1190 SBC shift_flag:BCC too_small
1200 STA result_power:LDX offset
1210 LDY #0
1220 .move_pairs_loop LDA result,X
1230 STA result,Y:INX:INY:CPY wlenp1
1240 BNE move_pairs_loop
1250 .MSD_filled:LDA result:AND #&F0
1260 BNE round_off:LDY #0
1270 JMP left_shift
1280 .left_loop2 LDA result,Y
1290 LSR A:LSR A:LSR A:LSR A:DEY
1300 ORA result,Y:STA result,Y:INY
1310 .left_shift LDA result,Y
1320 ASL A:ASL A:ASL A:ASL A
1330 STA result,Y:INY:CPY wlenp1
1340 BNE left_loop2:LDA result_power
1350 BEQ too_small:DEC result_power
1360 JMP left_justify
1370 .too_small LDX wlenm1:LDA #&80
1380 STA result+64:LDA #0:STA result+65
1390 .small_loop STA result,X:DEX
1400 BPL small_loop:RTS
1410 .round_off LDY wlen:LDA result,Y
1420 AND #&F0:CMP #&50:BCC finished
1430 SEC:DEY
1440 .round_loop LDA result,Y:ADC #0
1450 STA result,Y:DEY:BPL round_loop
1460 BCC finished
1470 JSR shift_result_right:LDA result
1480 ORA #&10:STA result
1490 LDA result_power:CMP #&FF
1500 BNE round_not_max:JMP max_power
1510 .round_not_max INC result_power
1520 .finished:LDA result_power
1530 STA result+64:LDA sign
1540 STA result+65:RTS
1550 .add_sub_init LDA first+64:SEC
1560 SBC second+64:STA power_diff
1570 BCS no_swap:JSR swap_first_second
1580 LDA first+64:SEC:SBC second+64
1590 STA power_diff
1600 .no_swap CMP wlent2:BCC add_cont1
1610 JMP dont_calc
1620 .add_cont1 LDA #0:STA shift_flag
1630 LDA first+64:STA result_power
1640 LDA power_diff:CLC:LSR A
1650 STA power_diff:BCC even
1660 DEC shift_flag
1670 .even LDY #0:LDX power_diff
1680 .load_loop LDA second,Y
1690 STA result,X:INY:INX:CPX wlenp1
1700 BEQ end_load:CPY wlen:BEQ end_load
1710 JMP load_loop
1720 .end_load BIT shift_flag
1730 BPL add_cont2
1740 JSR shift_result_right
1750 .add_cont2 RTS
1760 .max_power LDY wlen:LDA #&99
1770 .max_loop DEY:STA result,Y
1780 BPL max_loop:LDY wlen:LDA #&FE
1790 STA result,Y:INY:LDA operator
1800 CMP ASC("*"):BEQ max_mult_div
1810 CMP ASC("/"):BEQ max_mult_div
1820 LDA first,Y:STA result,Y:RTS
1830 .max_mult_div LDA first,Y
1840 EOR second,Y:STA result,Y:RTS
1850 .swap_first_second LDX #65
1860 .swap_loop LDA first,X:PHA
1870 LDA second,X:STA first,X:PLA
1880 STA second,X:DEX:BPL swap_loop
1890 LDA operator:CMP #ASC("-")
1900 BNE no_sign_change:LDA first+65
1910 EOR #&FF:STA first+65
1920 .no_sign_change RTS
1930 .dont_calc PLA:PLA:LDX wlenp1
1940 .dont_loop1:LDA first,X
1950 STA result,X:DEX
1960 BPL dont_loop1:RTS
2950 ]
2960 NEXT
2970 A$="SAVE FPcode "+STR$~code+" "+ST
R$~O%+" 6000 6000"
2980 PRINTA$:OSCLI A$
```

Listing 2

```
10 REM Program Arithmetic Demo
20 REM Version B1.0
30 REM Author  R.A. Beck
40 REM BEEBUG  March 1989
50 REM Program subject to copyright
```

```
  60 :
 100 MODE7:HIMEM=&6000
 110 X%=OPENIN"Fpcode":IF X% CLOSE#X%
 120 IF X%=0 PRINT"Please assemble mach
ine code before     running this demo.":
END
 130 *LOAD fpcode 6000
 140 INPUT"Number of significant figure
s (4 to 128) ";sig_figs
 150 IF sig_figs>10 THEN PRINT"No of si
g figs is too large for BASIC to handle"
 160 wlen=(sig_figs+1) DIV 2
 170 nlen=&6003:op=&6004
 180 first=&6005:second=&6047
 190 result=&6089:?nlen=wlen
 200 REPEAT
 210 INPUT '''"Enter first mantissa ";n
um1$
 220 INPUT "Enter first power ";power_i
nput1$
 230 INPUT '"Enter operator (+,-,*,/) "
;operator$
 240 INPUT '"Enter second mantissa ";nu
m2$
 250 INPUT "Enter second power ";power_
input2$
 260 PRINT''
 270 ?op=ASC(operator$)
 280 len1=LENnum1$
 290 len2=LENnum2$
 300 IF LEFT$(num1$,1)="-" THEN minus1=
TRUE:num1$=RIGHT$(num1$,len1-1):len1=len
1-1 ELSE minus1=FALSE
 310 IF LEFT$(num2$,1)="-" THEN minus2=
TRUE:num2$=RIGHT$(num2$,len2-1):len2=len
2-1 ELSE minus2=FALSE
 320 num1$=LEFT$(num1$,wlen*2)
 330 num2$=LEFT$(num2$,wlen*2)
 340 len_pow1=LENpower_input1$
 350 len_pow2=LENpower_input2$
 360 power1$=power_input1$
 370 power2$=power_input2$
 380 IF LEFT$(power1$,1)="-" THEN min_p
ow1=TRUE:power1$=RIGHT$(power1$,len_pow1
-1) ELSE min_pow1=FALSE
 390 IF LEFT$(power2$,1)="-" THEN min_p
ow2=TRUE:power2$=RIGHT$(power2$,len_pow2
-1) ELSE min_pow2=FALSE
 400 power1$=LEFT$(power1$,3)
 410 IF VAL(power1$)>126 THEN power1$="
126"
 420 power2$=LEFT$(power2$,3)
 430 IF VAL(power2$)>126 THEN power2$="
126"
 440 IF LENnum1$<wlen*2 THEN FORx=LENnu
m1$+1 TO wlen*2:num1$=num1$+"0":NEXT
 450 IF LENnum2$<wlen*2 THEN FORx=LENnu
m2$+1 TO wlen*2:num2$=num2$+"0":NEXT
 460 IF min_pow1 THEN first?64=&80-VAL(
power1$) ELSE first?64=VAL(power1$)+&80
 470 IF minus1 THEN first?65=&FF ELSE f
irst?65=0
 480 IF min_pow2 THEN second?64=&80-VAL
(power2$) ELSE second?64=VAL(power2$)+&8
0
 490 IF minus2 THEN second?65=&FF ELSE
second?65=0
 500 FORbyte=0 TO wlen-1
 510 first?byte=VAL(MID$(num1$,byte*2+2
,1))+VAL(MID$(num1$,byte*2+1,1))*&10
 520 second?byte=VAL(MID$(num2$,byte*2+
2,1))+VAL(MID$(num2$,byte*2+1,1))*&10
 530 NEXT
 540 CALL &6000
 550 PRINT'"Code answer  ";
 560 IF result?65>0 THEN PRINT"-";
 570 n=?result
 580 PRINT;~n DIV 16;
 590 PRINT;".";
 600 PRINT;~n AND &F;
 610 FORx=1 TO wlen-1
 620 n=result?x
 630 IFn<10 THEN PRINT;"0";
 640 PRINT;~n;
 650 NEXT
 660 PRINT" E ";
 670 PRINT;result?64-&80
 680 PRINT'
 690 beeb$=""
 700 IF minus1 THEN beeb$="-"
 710 num1$=LEFT$(num1$,10)
 720 beeb$=beeb$+LEFT$(num1$,1)+"."+RIG
HT$(num1$,LEN(num1$)-1)+"E"
 730 beeb$=beeb$+power_input1$
 740 beeb$=beeb$+" "+operator$+" "
 750 IF minus2 THEN beeb$=beeb$+"-"
 760 num2$=LEFT$(num2$,10)
 770 beeb$=beeb$+LEFT$(num2$,1)+"."+RIG
HT$(num2$,LEN(num2$)-1)+"E"
 780 beeb$=beeb$+power_input2$
 790 IF VAL(power_input1$)<37 AND VAL(p
ower_input2$)<37 THEN PRINT"Basic's answ
er ";beeb$;"  = "'EVAL(beeb$);
 800 @%=&1000A+wlen*2*256
 810 IF VAL(power_input1$)<37 AND VAL(p
ower_input2$)<37 THEN PRINT"   or    ";E
VAL(beeb$)
 820 @%=&90A
 830 PRINT'"Press any key":A=GET:CLS
 840 UNTIL FALSE
```

An Auto-Dating Utility

Peter Richards describes a utility to incorporate the current date and time into Basic programs and View files.

There have been several utilities for the Master 128 published in BEEBUG to date-stamp files (Vol.6 No.3 p.40, Vol.6 No.6 p.46, Vol.6 No.10 p.45), but none of them put the date into the listing of a Basic program. This routine will automatically insert the date and time into a suitable REM statement in Basic listings, or into a suitable comment line (command CO) in a View text file, each time either is saved. Although View can automatically pick up the current date on a Master (using |D), this is only displayed or printed. It is not saved explicitly as part of the file. The routine listed here relies on the Master's real-time clock or equivalent, and cannot be used otherwise.

To include a date and time in a program listing, the first line must be a REM statement with *at least* 18 characters after the REM keyword. Failure to achieve this will result in a corrupted program. A suitable View file should have the stored command CO on the very first line, and this line must also be at least 18 characters long (set up with spaces initially if you wish). When a SAVE command is given the routine checks to see if the current language is either Basic or View. If it is, and the first line is suitable, the date and time is written over the first 18 characters of the first line of the text, and then the file is saved in the usual way. If the checks fail then no action is taken apart from saving the file.

Type the program in and save it. Running the program will produce a file called DATE containing the assembled code for the utility. As given, it assembles the code to run in the transient utility workspace at &DD00. This is safe provided the command *MOVE is not used in a shadow screen mode. Alternatively, it could be assembled to run at, say, &900. The code produced is 256 bytes long, and just fits into one page of memory.

Once the object code has been assembled, typing *DATE will install the routine. The routine will work with any filing system and with a turbo co-processor. However, it is necessary to type *GO DD00 or re-install the routine after changing filing systems as the vectors are reset when the change is made. When a turbo co-processor is used it is also necessary to re-install the routine each time a language change is made.

PROGRAM DESCRIPTION

Lines 150-190 set up the interception of the vector to the OSFILE routine.

Line 210 branches to the end of the routine if the OSFILE call is not a SAVE.

Lines 220-240 save the registers and 4 zero page locations.

Lines 250-280 save the OSFILE parameter block which would otherwise be corrupted.

Lines 290-320 read the date and time.

Lines 330-340 read OSHWM.

Lines 350-380 read the current language and branch accordingly.

Lines 400-460 restore all the saved locations.

Lines 500-550 check the first line of a Basic program.

Lines 570-690 check the first line of the View text. (The text starts at OSHWM+&101).

Lines 710-750 write the date and time into the first line, having jumped forward the correct number of bytes.

Lines 760-810 write spaces at each end of the date and time.

AUTO-DATE AND CONFIRM

An extended version of this program is also included on this month's magazine disc/tape.

This combines the new date/time stamp routine with Bernard Hill's Auto-confirm utility (BEEBUG Vol.6 No.6). Again the program must be run to assemble the code (410 bytes), which may then be installed by typing *CONDAT.

```
   10 REM Program DATER
   20 REM Version B1.1
   30 REM Author  P.M.Richards
   40 REM BEEBUG  March 1989
   50 REM Program subject to copyright
   60 :
  100 osbyte=&FFF4:osword=&FFF1
  110 FOR pass%=0 TO 3 STEP 3
  120 P%=&DD00:Q%=P%
  130 VDU15
  140 [OPT pass%
  150 LDA &212:STA oldfilev
  160 LDA &213:STA oldfilev+1
  170 LDA #filev MOD 256:STA &212
  180 LDA #filev DIV 256:STA &213
  190 RTS
  200 .filev
  210 CMP #0:BNE okfilev
  220 PHX:PHY
  230 LDA &A8:PHA:LDA &A9:PHA
  240 LDA &AA:PHA:LDA &AB:PHA
  250 STX &A8:STY &A9
  260 LDY #17
  270 .loop5
  280 LDA (&A8),Y:PHA:DEY:BPL loop5
  290 LDX #block MOD 256
  300 LDY #block DIV 256
  310 LDA #0:STA block:LDA #14
  320 JSR osword
  330 LDA #131:LDX #0:LDY #255
  340 JSR osbyte:STX &AA:STY &AB
  350 LDA #252:LDX #0:LDY #255
  360 JSR osbyte
  370 CPX #12:BEQ basic
  380 CPX #14:BEQ view
  390 .exit
  400 LDY #0
  410 .loop6
  420 PLA:STA (&A8),Y:INY
  430 CPY #18:BNE loop6
  440 PLA:STA &AB:PLA:STA &AA
  450 PLA:STA &A9:PLA:STA &A8
  460 PLY:PLX:LDA #0
  470 .okfilev
  480 JMP(oldfilev)
  490 .basic
  500 LDY #4:LDA (&AA),Y
  510 CMP #&F4:BNE exit
  520 DEY:LDA (&AA),Y
  530 CMP #23:BCC exit
  540 LDX #5:JSR write
  550 BRA exit
  560 .view
  570 INC &AA:INC &AB
  580 LDY #0:LDA (&AA),Y
  590 CMP #&80:BNE exit
  600 INY:LDA (&AA),Y
  610 CMP #ASC"C":BNE exit
  620 INY:LDA (&AA),Y
  630 CMP #ASC"O":BNE exit
  640 .loop2
  650 INY:LDA (&AA),Y
  660 CMP #&D:BNE loop2
  670 CPY #21:BCC exit
  680 LDX #3:JSR write
  690 BRA exit
  700 .write
  710 CLC:TXA:ADC &AA:STA &AA
  720 LDY #17
  730 .loop
  740 LDA block+3,Y:STA (&AA),Y
  750 DEY:BNE loop
  760 LDA #32
  770 STA (&AA)
  780 LDY #12:STA (&AA),Y
  790 LDY #18:
  800 LDA (&AA),Y:CMP #&D:BEQ skip
  810 LDA #32:STA (&AA),Y
  820 .skip
  830 RTS
  840 .block
  850 EQUS STRING$(24," ")+CHR$13
  860 .oldfilev
  870 EQUW 0
  880 ]
  890 NEXT pass%
  900 PRINT'P%-Q%" Bytes"
  910 OSCLI("SAVE DATE "+STR$~Q%+" "+STR
$~P%)
  920 PRINT'"The assembled code has been
 saved to"'"disc under the filename 'DAT
E'"'
  930 END
```

♣♦♥♠ Colossus Bridge 4 ♠♥♦♣

Reviewed by Sheridan Williams

Product	Colossus Bridge 4
Supplier	CDS Software Ltd CDS House, Beckett Road, Doncaster DN2 4AD. Tel. (0302) 321134
Price	£14.99 (5.25" disc), £11.99 (tape), inc. VAT

Colossus Bridge 4 comes on 40-track DFS disc, or a tape, and is accompanied by the paperback book "Begin Bridge" by G.C.H. Fox.

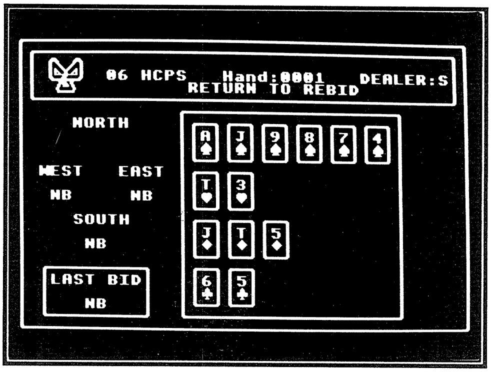
Unlike Chess, Bridge seems to suffer a dearth of good computer versions of the game. "Colossus Bridge 4" remedies this situation, and is available for the Spectrum, Commodore, Amstrad, PC compatibles (with CGA card), the BBC Model B, B+, Master, Compact and Electron. The package really is jolly good, and I would encourage you to read on beyond some of my more negative initial comments.

Unfortunately BBC and even Master users have machines with insufficient memory to make available features such as saving and loading a game, backstepping the play, demonstration mode, on-line instruction menu, and listing the deal to a printer. Nevertheless several useful features remain:

During the Play:	Autoplay a card Claim remaining tricks Recommend a card Abandon the hand
At end of Game:	Replay a hand Input a hand Start a new rubber Set distribution of hand Set the playing speed

CDS Software says: "Colossus allows one player to play a complete game of Bridge with the computer controlling the other three hands. Each hand is bid according to the Acol system, with Blackwood, Stayman, and Baron conventions being used. The strong two club, and take-out doubles are also supported. All four hands are displayed at the end of play."

So how does the package live up to expectations? Well, judging by the manual (not the free book "Begin Bridge", of which more later) you would be rather disappointed. The manual is tiny, measuring only 100mm x 130mm, and contains only 13 pages. It tries to cover all the features on all the computers listed earlier. Although typeset, only one typeface and font size is used, and this tends to make headings and other important features get lost, resulting in a pretty difficult-to-follow manual. Amazingly, once you have used a "highlight" pen to mark headings and relevant sections, the manual is quite a useful summary if treated like reference card.

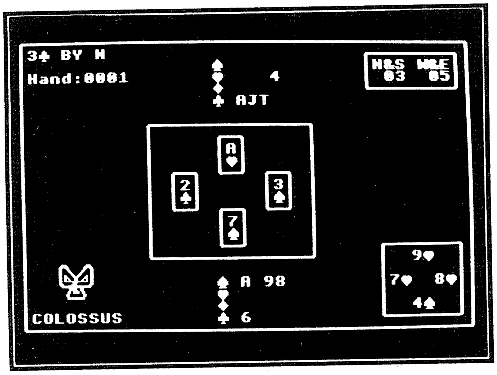


Initial bidding

♥ THE BOOK

CDS has, very wisely, chosen to supply the book "Begin Bridge" by G.C.H. Fox to augment the manual, reasoning that there is no point in their manual attempting to teach you how to play when there have been suitable books around for ages. I would concur with The

Observer's Terence Reese when he says "A clear and reliable introduction for the complete beginner." although I would go further and say that it caters for more than the beginner, and is an excellent book to read if your Bridge is a little rusty. Pity it has no index.



Game in progress

♥ THE PROGRAM

On most computers other than the BBC/Master range, the package allows you to change the screen colours. Unfortunately, the standard colours make playing on a green screen monochrome set virtually impossible, whereas on a colour screen I found the colours pretty much ideal. Up to now I may have sounded very negative about the package, but as I said earlier you should read on.

Bridge is divided into two parts - the bidding and the play: each requiring skills in its own right. If anything, I personally prefer the bidding to the play. It involves a quite separate challenge of communicating with your partner, listening to the opponents bids, and weighing up the statistical evidence to arrive at a contract. Although you may not know it until after the play, it is most satisfying if you have bid the best contract. Colossus Bridge caters well for those who like bidding.

Once the package has been auto-booted into action, you are prompted for a playing speed from 0 to 28. This sets the delay between each bid, and also the delay between each card being played. I found that 15 was appropriate in most instances. The package makes abundant use of the space bar, the moral being "if in doubt press space".

♥ THE BIDDING

The current state of vulnerability, the dealer, your high-card point count (face cards only, not distribution), current score (below the line), and hand number, are displayed at the top of the screen. Being South your hand is displayed at the bottom of the screen, you are always South, this cannot be changed. The bidding is displayed at the left of the screen with the selected delay between bids. It is very easy to enter a bid, the convention being fairly natural, 1 Spade is 1S, 4 Clubs is 4C, 3 No Trumps is 3N, Double is D, Redouble is R, and Pass is the ubiquitous space bar. Unless specified at the end of each game, the next hand is randomly generated.

As has been said before bidding follows the Acol system, with Blackwood, Stayman, and Baron conventions. Within reason you can trust the computer's bids, whether it is bidding as your partner or as the opposition. Its bidding is defined in the manual, however it does get confused on some very unusual hands, but in general it won't make beginner's mistakes such as passing a 2-club opener, or opening with the lower ranking of two suits of equal length (and for those of you who know - yes, it does know what to do if the suits are spades and clubs, or it has a 4441 distribution).

♥ THE PLAY

As you would expect, in some hands North/South end up in a contract, in others East/West end in a contract and you are in opposition. Either way it is good to think for a while before playing a card, as once chosen, you cannot retract the card on the BBC/Master as you can on other computers. However, on the Master you can ask the computer to recommend a card. The computer plays well,

Continued on page 60

512 Forum

By Robin Burton

After the single topic of the last issue I thought we'd have a change, with rather a mixed bag this month.

LETTERS

Thanks again for your letters; they provide both inspiration and information. The good news is that because of your enthusiasm 512 Forum is to become a regular column. Unlike some publications, BEEBUG does respond directly to its readers, so if you want the Forum to continue keep the letters coming. I was pleasantly surprised by the number in January, but this means it's time for some guide-lines.

Inevitably, some letters relate to highly specific points which may not be suitable for general inclusion in the Forum. That's OK, I'll try to reply to letters personally in these cases, but you please do quote your membership number in all correspondence, while an SAE does help if you want a direct reply.

SOFTWARE COMPATIBILITY

I'll try to answer queries, but clearly I don't know every package. Several readers have again raised the idea of a software compatibility list. Although I've mentioned this before, let's see what can be done.

Acorn have for some time compiled a list of known software and the latest version will be included in the Dabs 512 User Guide (to be published shortly). The trouble is that such a list is never complete. This is where you come in. If you have definite knowledge that a package does or doesn't run, or you know what's needed to make one work that otherwise doesn't, let me know about it. As the data accumulates we'll publish updates and additions in the Forum.

This can continue indefinitely, but it's up to each of you to pass on your knowledge for the benefit of fellow 512 users. My view is this - apart from 512 support by BEEBUG and Dabs Press we're virtually on our own, so we have to help each other. Over to you.

DOCUMENTATION AGAIN

Next, updates on items from December's Forum. Thanks are due to both G.R. Smith for his letter and to Richard Grant for his extra efforts on your behalf in clarifying the situation on Digital Research manuals.

The 'official' line, as stated in December, had been checked with Digital, but, it seems that a different result is possible if your enquiry is unofficial or private. Richard suggests, however, that you may need to be fairly persistent (and ideally speak to a lady called Patricia - won't she be pleased?).

Although DOS+ is not a current product and therefore is no longer officially supported by DR, they can supply manuals after all. The user guide for example, is £17.50 in loose sheet A5 form, plus £6.50 for the optional hard cover ring binder.

Programmers References are also available, but at rather higher cost, since there are two parts, one for CP/M and one for DOS Plus. These will only be of interest if you write your own code, and the contents are exactly the same as the Glentop book mentioned previously (BEEBUG Vol.7 No.7), so check the prices carefully. A System Reference Guide is also available, again at higher cost, but be warned this is intended for system implementers and as such is of even more limited interest, even to programmers.

Having kindly been lent by Acorn the Winchester on which all the 512 software was developed, I can also tell those of you that do program that several standard documented CP/M and DOS+ interrupts have been deleted from the 512 implementation, by no means for obvious reasons in some cases. If you write your own code don't be surprised if some of the documented interrupts don't work! By the way, if you didn't understand these comments you don't need the last three manuals.

To pursue these manuals phone Digital Research on Newbury (0635) 35304 and ask for Customer Support. I'm told they take credit cards, so payment is simple.

PC+ UPDATE

Now for some other news, unfortunately not so good. I have again checked with Solidisk, and the latest on the PC+ is that the last batch consisted of 25 units and that they have no plans to produce any more. These were priced at about £120 because of the small quantity, and all had been sold within a fortnight. It looks like that's the end of the matter as far as the PC+ is concerned.

The reasons given are the current high cost of RAM - STL's existing RAM stocks are exhausted and new supplies would now cost twice as much, meaning that the PC+ would retail at over £200. In addition, STL are moving out of the Acorn market, and I've been told that they have no further interest in Acorn add-ons. It's a pity: although I don't have a PC+, I do know that it is well made and works reliably.

It looks like the only way you'll be able to expand your 512 to a megabyte from now on will be to build the memory extension detailed in the Dabs 512 Technical Guide.

STAR

I must confess to a mistake in the December issue, concerning the Dabs 512 utilities disc. The disc catalogue program 'DTYPE' allows you to catalogue DOS+ discs from ADFS, not the the other way round. Sorry, I obviously wasn't working very well that day because I didn't notice at the time.

There is no problem cataloguing an ADFS or DFS disc from DOS+ by means of the 'STAR' command, as many of you will know, but this provides an excuse to cover a topic that has been raised in several letters. Specifically, why is it that programs like BBC screen dumps or other routines can't be used, or crash the system if called from DOS.

First, if you're not too familiar with 'STAR' just try the following. Start at the normal DOS prompt, with a disc containing 'STAR.CMD' in drive A. Enter 'STAR' and press return.

You will find the screen clears and a '*' prompt replaces the DOS prompt, appearing near the bottom of the screen. You have now connected yourself directly to the BBC's OS command line interpreter, and can issue many BBC star commands from DOS, including some filing system commands. Be careful though, you can't do everything.

Try putting an ADFS disc in drive B and then try the following commands, pressing Return after each one:

```
mount 1
cat
in.*
map
free
```

You'll see that the disc is accessible using completely normal ADFS commands. You can also change directory, 'type' or 'dump' files and so on. Note that you needn't enter a star before each command, but if you do so out of habit it's harmless. To exit to DOS just press Return without an entry.

DFS discs can be accessed in a similar way (on the Master) by means of the temporary filing system. You can also issue FX calls and theoretically other * commands to ROMs in the host machine. If you decide to experiment a bit, don't do it if there's data in the 512, because much of the BBC's software will crash the system.

The thing you must avoid is altering either the host's or the 512's memory. For example '*LOADing' a BBC file will transfer the file contents to the 512's memory, not the BBC's, unless the file's load address is prefixed by &FFFF. Loading a BBC file will therefore corrupt DOS, even if it's not instantly obvious.

If you want to see how delicate things can be when you start experimenting, enter 'STAR FX13,4'. Be warned, it's the last thing you'll do before re-booting the system. Try it and see.

*FX13,4 turns off event 4, the screen vertical synchronisation event in the BBC. In an otherwise 'idle' machine, event 4 isn't even enabled normally. It is for DOS though, and if you disable it you'll instantly paralyse the entire system.

The reason for this simple but dramatic demonstration is to warn you against making assumptions about what you can do. The cost could be hours of work if you attempt something unusual or untested with data in the 512.

The point is that most BBC service ROMs and all languages must alter various OS settings or vectors in the host to suit themselves or to carry out their functions, and many also expect to be able to use memory too. DOS+ is no different, so these actions, if carried out by other software, crash the system when DOS is active. Here's why.

First, booting DOS always turns shadow RAM off, unless you have a non-standard shadow set-up in a model B. Of course starting DOS with a hard break normally turns shadow off in these cases (if you insist that shadow stays on you only get a blank screen in DOS anyway, not much use!).

Also, DOS+ intercepts four vectors in the BBC, generates two unknown events, and changes several other MOS settings too. Running the tube and disc software uses practically all the BBC's available RAM including that reserved for screen display (&3000 upwards).

I haven't bothered to work it out, but I'd guess there are in total perhaps about 500 to 1000 unused bytes in the BBC host when the 512 is running. The unused bytes aren't all together in one place either.

It's easy to see that if BBC routines can't use main RAM, can't intercept vectors and may neither enable nor disable events there's really not much they can do. If DOS+ is to continue to run, software must be specially and very 'legally' written to permit this. The simple fact is that most native BBC software isn't written that way, so for example preventing the screen clearing in 'STAR' would not be enough to allow, say, a BBC screen dump to work.

Restrict 'STAR' to things like 'FX6' to enable or disable printer line feeds, 'TV0,1' followed by 'pcscreen' to enable or disable interlace, or 'FX5' if you switch between serial and parallel printers. Used like this you'll have no problems. Try something more adventurous and you might well run into trouble.

'STAR' can also be issued as a composite command with a tail for one-off operations. For example, as the FX13 is above, or to stop line feeds being sent to the printer you could enter:

```
        star fx 6,13
```

Used for single commands like this 'STAR' is automatically followed by an immediate return to the DOS prompt.

CLOSE-DOWN

I'll close with a simple but neat idea from Gordon Sweet for Master owners, also using 'STAR'. Having cancelled the micro's line feed in DOS by using 'STAR' in his start-up 'AUTOEXEC.BAT' file, Gordon has gone a step further, automating the exit from DOS with another batch file, like this:

```
        star
        fx6,0
        configure notube
        echo Now press CTRL/BRK
```

Call the file 'FINISH.BAT' (or anything suitable, but NOT 'exit', that's a DOS command) then to leave DOS simply enter the command - 'finish', pressing Ctrl-Break when instructed. The batch file can also contain a 'BYE' to park hard disc heads if required and as Gordon says, can include other *configure commands (e.g. to select DFS instead of ADFS, internal/external tube etc) all of which come into effect immediately you carry out the hard-break. B

The Ins and Outs of Basic (Part 3)

To conclude our look at input and output techniques in Basic, Mike Williams devotes most of this month's instalment to the use of the INKEY function.

To finish our examination of the different methods of input and output available to the Basic programmer, I want to deal here with the INKEY function. INKEY is very much like the GET function, which we looked at last time, but instead of waiting indefinitely for a keyboard response, it waits only for a predetermined period of time. There is also a further variation known as the negative INKEY function which we will deal with later in this article.

As your User Guide will tell you, the basic format for INKEY is:

```
code=INKEY(n)
```

or:

```
char$=INKEY$(n)
```

In both cases, 'n' specifies how long the function should wait for a response. This time is measured in *centi-seconds*, so INKEY(100), for example, would wait exactly 1 second. Like GET and GET$, discussed last month, INKEY will return the ASCII code of a character entered, while INKEY$ will return the character itself. If no character is available within the specified time, then INKEY returns a value of -1, and INKEY$ a null string ("").

TIME DELAYS

Many programs require the use of a time delay. For example, some games programs are too fast to play unless slowed down. It might be thought that an obvious choice for implementing such a delay would be to use the INKEY function. The disadvantage of this is that it is vulnerable to an accidental, or even deliberate, key press.

As ideas on incorporating delays into programs can be quite important, we shall allow ourselves a short diversion to consider other ways of handling delays, as I believe the ideas are relevant, and moreover useful. Since the purpose of a delay is simply to provide a pause before continuing, rather than to obtain any input, a better technique is to use the *TIME* function, for example:

```
t=TIME:REPEAT UNTIL TIME-t>n
```

where n is the length of the required delay, again in centi-seconds. This is also much better than using an empty FOR-NEXT loop, such as:

```
FOR I=1 TO 1000:NEXT
```

True, this is also impervious to any keyboard input, but the length of the delay will depend upon the version of Basic and on which machine it is running. Using this, a program which is just right on a model B may be too quick on a Master. It also gives no indication of the length of the delay being used, and therefore adjustment is more a hit and miss affair. All in all it is much better to use TIME for any fixed length delay.

There is one possible advantage to the FOR-NEXT technique. With INKEY or TIME, the smallest unit of time is one centi-second. With the FOR-NEXT approach, smaller variations in timing can be achieved, useful if you need it and the disadvantages are less important to you.

One further point to note with regard to timing is that the instructions in any Basic program take a finite time to be executed. Suppose you want to write a routine that will display a countdown in seconds from some initial total. Here is a possible procedure to do this.

```
1000 DEF PROCcountdown(n):LOCAL t
1010 REPEAT
1020 PROCdisplay(n)
1030 t=TIME:REPEAT UNTIL TIME-t>100
1040 n=n-1
1050 UNTIL n=0
1060 PROCdisplay(0)
1070 ENDPROC

1100 DEF PROCdisplay(n)
1140 PRINTTAB(0,11) CHR$141,n
1150 PRINTTAB(0,12) CHR$141,n
1160 ENDPROC
```

The countdown procedure uses a REPEAT-UNTIL loop to count down one second at a time. The actual timing is done by a further REPEAT-UNTIL loop using the TIME variable as already described. Each time we execute the main loop, the seconds remaining are displayed on the screen, and this value reduced by one until zero is reached.

For convenience, the actual display is performed by a second procedure which assumes that mode 7 is in use to display the seconds count in double-height figures (we'll be looking at a way of producing a much larger display in next month's First Course).

Now let's try out these procedures with a short program:

```
100 MODE 7
110 REPEAT:CLS
120 INPUT"Time: " t
130 PROCcountdown(t)
140 G=GET
150 UNTIL FALSE
160 END
```

The program will prompt for a value - try 20, and you should see the screen value count down from 20 to 0.

But how long does this take? You could try timing it with your watch but that might be difficult, so let's get the program to do it for us. just add the two lines:

```
125 TIME=0
135 PRINT TIME/100
```

Now run the program again. Whatever number of seconds you specify, the countdown (as timed by the computer) will take slightly longer than it should. For example, running on a Master, a countdown of 20 seconds was timed at 20.83 seconds. This is simply because of the time taken to execute the instructions in the program.

To get a more accurate result, we need to change the delay time. I found that 96 gave a better result (change line 1030), but you will need to experiment to find the best value on your own system (you are looking for the value which gives a final result as near as possible to the countdown time you type in).

The point is simply that if timing in a program is really critical, then the time to execute the instructions in the program must also be taken into account. But do make sure you don't spend more time and effort than the application deserves. Likewise, you might argue that the instructions to check the countdown timing will also take a finite time to execute and will therefore effect the result. Well so they might, but the effect of just two instructions is likely to be negligible, otherwise we'd go on for ever.

TIMED INPUT

You might now begin to question whether INKEY has a role at all. Well it certainly does. For a start you may genuinely want to incorporate a time delay into a program, but still allow the user the option to continue sooner by pressing an appropriate key.

The most likely situation for using the INKEY function is within a loop where we are expecting to receive some input from the user, but at the same time we may want to continually update or change information already on the screen. This is often the case in a game-playing situation where the user controls one character on the screen, but other characters may move independently. Another example might be a database or similar application where we want to display a continually updated clock on the screen while still waiting for user input.

A typical loop to achieve this would take the format:

```
REPEAT
key=INKEY(n)
IF key>-1 THEN PROCuserinput(key)
PROCupdatescreen
UNTIL FALSE
```

The value for n will depend upon the circumstances, but 1 or 2 would be quite typical, even 0 - after all the object is usually to try and maintain some impression of continuous movement or change on the screen, so the smaller the delay the better.

If user input does take place, then INKEY returns a positive value and hence execute PROCuserinput with the key entered as its parameter. If no key is pressed, INKEY will return a '-1' and so only PROCupdatescreen will be executed to do whatever is needed anyway. This procedure could be executed every time a pass is made round the loop (as here), or only if no user input occurs.

There is one problem which may occur in these circumstances, and to understand it it is necessary to look a little more deeply into what goes on when keyboard input takes place. Whenever a key is pressed (with a few exceptions), the corresponding ASCII code is stored in a reserved area of memory called the *keyboard input buffer*. When any kind of input instruction is executed, and this includes INKEY, Basic looks not at the keyboard but at the keyboard input buffer, and takes the next character from there.

If auto-repeat is in operation (as it usually is), holding down any key for even a fairly short period of time will begin to fill up the input buffer with characters. If it is the INKEY function that is being executed, then any timing built into the function will be lost as every time INKEY is executed a new character will be ready and waiting in the input buffer.

Now this may be perfectly acceptable, even desirable. Where the result of INKEY is being used to move an object (perhaps a cursor in a graphics program, or a character in a game), then repeated pressing of a key, and the over-riding of the INKEY delay which follows, may be essential in achieving smooth or rapid movement.

However, that is not always the case, and it may be essential to ensure that there are no characters waiting in the input buffer when an INKEY instruction is executed. This can be achieved with an FX call, the best one to use being *FX15. In fact, the BBC micro's memory is organised into a number of different buffers each with a separate function. In particular, *FX15,1 will *flush* the current input buffer. Our skeleton loop from before would then become:

```
REPEAT
*FX15,1
key=INKEY(n)
IF key>-1 THEN PROCuserinput(key)
PROCupdatescreen
UNTIL FALSE
```

There are many situations when this FX call is useful, as it clears any existing and unwanted information out of the keyboard buffer. This is particularly useful when auto-repeat may have caused more data to enter this buffer than had been realised by the user.

NEGATIVE INKEY

In addition to what has already been said there is one further and particular use of the INKEY function where a negative value is specified (hence the usual name for this variation). In these circumstances, the negative value has no time related meaning at all, but refers to a key on the keyboard. Furthermore, all the keys on the keyboard can be referenced in this way, including those such as Shift and Ctrl which do not by themselves generate ASCII values at all.

When Basic executes such a function, it does not look at the keyboard input buffer at all. Instead it looks directly at the specified key to see if it is depressed at that instant. If it is a value of TRUE (-1) is returned, otherwise FALSE (0) is returned. One further advantage of the negative INKEY function is that it is comparatively fast in operation, an important factor in, for example, game playing programs.

Negative INKEY functions can be used in a loop to check for input. For example:

```
REPEAT
IF INKEY-58 THEN PROCup
IF INKEY-42 THEN PROCdown
IF INKEY-26 THEN PROCleft
IF INKEY-122 THEN PROCright
PROCupdatescreen
UNTIL FALSE
```

will check for any of the four cursor keys being pressed, and will respond correctly if more than one key is pressed at the same time (whether that has any sensible meaning is up to the program and programmer). Each and every possible keypress will now have to be tested for

Continued on page 60

ShareVAL - An Investment Analysis Package

Reviewed by Mark Jolliffe

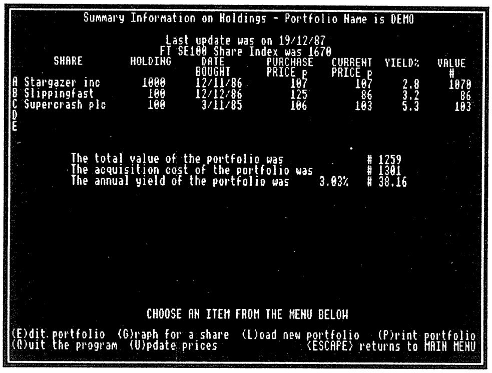
Product	ShareVAL
Supplier	Nasco Software 75 Maltese Road, Chelmsford, Essex CM1 2PB. Tel. (0245) 251498
Price	£16.95 inc.

ShareVAL is described as "a suite of programs to help you to keep track of your investments in shares, unit trusts and building societies". It is usable on the BBC B, B+, and Master with at least one disc drives, and uses the Acorn DFS format. The programs are supplied on a 5.25" disc, with an effectively presented, if cheaply printed, instruction booklet. The review was conducted on a Master Turbo, the presence of the second processor giving no advantage other than screen update speed. A printer is an advantage, but is not essential.

ShareVAL is intended to record the date of purchase and cost of a share, and its present price and yield, on a weekly basis over a period of up to ten years. From these details a portfolio valuation table is produced, and three graph options: share price with FTSE 100, portfolio valuation with cost, and portfolio yield. A single 40 track drive can accommodate one portfolio of fifteen shares, while further portfolios can be created if an 80 track and/or a second drive is available. All the portfolios share a common start date, which is set by the !BOOT file, together with the number of the data disc drive and the default portfolio name.

The programs are menu driven, either as numbers from mode 7 screens, or the initial letters of options from the mode 0 sections. The former is used for the main selections, whilst the latter is used for reports, editing, portfolio creation, and data input.

When booted, a title screen in mode 7 appears followed by a description of the program. If a printer is not connected a warning of this then appears. Another screen asks for the date (automatic on a Master), followed by a request for the FTSE 100 index for the day. Each screen is cleared by a singularly tedious blue curtain flood. Garish in colour, and irritating to the eye, this feature has nothing to commend it, and is quite out of place in this type of program. Equally, having to key through the introductory screens each time the program is run is time wasting and gradually gets more and more annoying.



```
        Summary Information on Holdings - Portfolio Name is DEMO

                      Last update was on 19/12/87
                     FT SE100 Share Index was 1670
     SHARE         HOLDING     DATE        PURCHASE   CURRENT  YIELD%    VALUE
                              BOUGHT       PRICE p    PRICE p               #
A Stargazer inc      1000    12/11/86         107        107     2.8     1070
B Slippingfast        100    12/12/86         125         86     3.2       86
C Supercrash plc      100     3/11/85         106        103     5.3      103
D
E


        The total value of the portfolio was                # 1259
        The acquisition cost of the portfolio was           # 1301
        The annual yield of the portfolio was     3.03%     # 38.16




                    CHOOSE AN ITEM FROM THE MENU BELOW
(E)dit portfolio  (G)raph for a share  (L)oad new portfolio   (P)rint portfolio
(Q)uit the program  (U)pdate prices                 <ESCAPE> returns to MAIN MENU
```

Sample portfolio

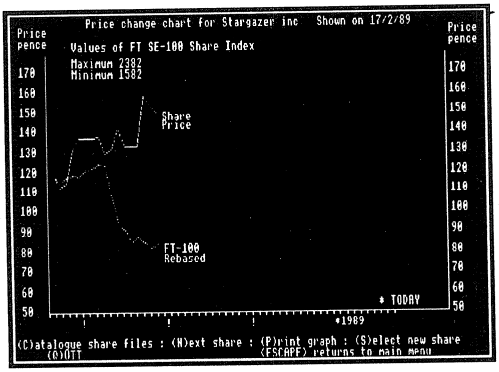
The mode 0 sections are far better, and will be much more acceptable to monochrome monitor users. Serious users of this type of program will be far more impressed with clarity of presentation than with the gratuitous use of meaningless colour, and will certainly not need to be told what the program is all about every time it is run.

The program is supplied with a demonstration file of fictional share details, which appears as the default portfolio. The main menu gives the opportunity to switch to a different portfolio, go to the "portfolio table", produce any of three graphs, or go to 'Utilities'.

The portfolio table is a mode 0 list of portfolio details such as purchase date and price, current price and yield, and value. Below this is given the cost of the portfolio, what its value was, what its yield was, and the value of that yield. At the bottom of the screen is a simple menu of options to edit any of the details, create a new portfolio, update prices, or produce a printout.

The treatment of yield is, to say the least, misleading. For a share or unit trust, the yield is the relationship between the last declared dividend and the present price. As such it is just a forecast, and can only give a guide to future return. For a building society account, treating the interest rate as a yield (as the instructions suggest) ignores rate variation, compounding, or whether interest is earned daily, weekly, monthly, quarterly, or whatever.

The program simply takes the present percentage yield for each share or account, works this out in pounds, adds them all up, and then works backwards to give the overall yield as a percentage and amount, and presents this as if it has actually happened. Total fiction! The yield is a valid variable to record and graph, but in this case has been pushed too far. It would have been much better if actual dividends or interest could have been entered, from which the true yield could have been obtained.


Price change chart for Stargazer inc   Shown on 17/2/89
Price pence
Values of FT SE-100 Share Index
Maximum 2382
Minimum 1582
170
160
150
140
130
120
110
100
90
80
70
60
50
Share Price
FT-100 Rebased
* TODAY
*1989
(C)atalogue share files : (N)ext share : (P)rint graph : (S)elect new share
(Q)UIT
(ESCAPE) returns to main menu

Sample price change chart

Price entry or editing of details is straightforward, but the confirmation prompt has a vestigial mode 7 character 134 in front - a legacy from an earlier version which should have been removed by now. Price entry is integer only, which might annoy some. The pound sign has also been replaced by the hash on screen - a lazy way of getting the pound sign to print out properly on most printers.

The graph options use mode 0 and are adequate. Scaling is automatic - or at least, it cannot be altered. The horizontal time scale is bereft of detail, and is selected as between one and five years back from the present. Graphing beyond five years requires the program to be re-run to enter a bogus, historical, date, instead of the present one. One possibly confusing feature concerns the plotting of the FTSE 100 index re-based to each share. Since the FTSE is the standard by which the others are judged, it would perhaps have been better if it had been left alone, and the others been re-based to it. A choice would have been better still.

The Utilities give the opportunity to view or change historical details, via a crude *CAT display. Share names are truncated to seven characters, or up to a space, for catalogue use, whilst remaining full length elsewhere. "GUS A" and "GUS B" both become "GUS", and similarly both "United Biscuits" and "United Newspapers" become "United". In each case, both are treated as the same share and prices of the second overwrite the first, but both feature in the portfolio table as separate entries.

Failure to enter a valid "short filename" dumps one unceremoniously out of the program via an untrapped error, as also happens when the file catalogue is full. Historical data can be entered through this editing route, but it is no more convenient than re-running the program with a new date for each, a daunting task. No utility exists to alter historical data to bring it into line after Rights or Scrip issues, or instalment payments such as with privatisation stocks. Graphing such unaltered data can give a misleading indication of performance.

The program could be made to work well: with the yield treatment corrected it could well find a use with the casually interested investor, or in the junior school, where neither price nor expectations are high. The impression given is of a lightweight, rather than serious, application. As such, it should not be compared with packages costing many times as much, directed at the serious investor. Its main failing is in over-reaching itself, being flawed in a feature superfluous to the budget end of the market at the same time as having features irritating to the more demanding user.

BEEBUG Education

by Mark Sealey

Mark Sealey offers a further personal response to the impact of the new National Curriculum on the educational use of computers, and explains how users of Acorn equipment are particularly well placed to respond to this challenge.

It may seem perverse, having been so critical of the whole idea of a National Curriculum, to return to the subject this month. BEEBUG Education does so for two reasons. Firstly because unfortunately, there is no going back: as of September this year, all of us involved in teaching with (and without) micros will be obliged to attend to it in some way. Secondly, the third and final Government document referring to the work with IT has now appeared in its interim form,and it will affect the teaching of all of us in one way or another.

The report on Design and Technology was published at the end of last year. It follows the format of the others in its concentration on testing and assessment. Yet it is not explicit about the consequences where pupils fail to gain one of the notional assessment targets. Regrettably, though, this report too follows a prescriptive, knowledge-based approach to Technology and IT. It is widely feared that the curricular lead which Britain has achieved for innovative work in educational computing may be lost as these new targets are pursued.

It is beyond the scope of these pages to examine ways in which imaginative teachers can offset the damage until the National Curriculum in its present form at least is eventually abandoned as unworkable - as it one day surely must be. Nor would it be practical to go into detail about all the report's omissions. Some such areas may indeed be included when it appears in its final form in the late spring.

Unlike the Mathematics and Science proposals dealt with in BEEBUG Education in BEEBUG Vol.7 No.7, the Design and Technology report has a separate chapter on *IT and Computing*. It does not, though, deal any more fully with the cross-curricular use of IT than the former did. Although the brief was for each of the working parties to investigate the role of IT in their curricular areas, no proper attempt at integration yet looks likely.

The report recognises the important role played by pupils' own experiences of computers, and even explicitly acknowledges [para 3.3] that such variety of experience is an asset. Not only are many children and students likely to own a BBC micro, Master or Compact, and use them at home for games for instance, but the vast majority will be familiar with Acorn equipment from school or college. So although the report does not mention makes or machines, its contents are highly relevant to these pages in BEEBUG just the same.

Use of IT is divided into two strands in chapter 3 of this publication. IT is to be seen in the first place as a support for such activities as word processing, modelling, and graphic design, where computers are essentially tools. Secondly, it lies at the heart of many of the systems in use around us: from point of sale complexes, through mechanical applications like lathe-cutting systems to viewdata displays. You may like to consider whether this is a realistic and helpful division.

Early on in the section [para 3.5], attention is drawn to the need to encourage pupils to understand general principles in preference to particularities. This is to be applauded. Schools, colleges and students who have stuck with Acorn equipment are well served here. The way the model B and Master were designed has always offered easy access to their operating systems, and a good variety of programming languages is also available.

There is one important result to this. It is the extent to which pupils have consequently been able to derive particularly rich insight into generalities - by comparing how two different languages attain their objectives at high as well as at machine-level, for example. At the same time, the availability of software like the 6502 Developers Toolbox, as well as more general products like Disk Doctor and similar utilities, has meant that peering into the workings of the computer and its peripherals has not been as off-putting as it might.

There is much published material on the working of CP/M and MSDOS. These are the operating systems running on what were for a long time, and to some extent still are, Acorn's chief rivals in the education market (the RML 480Z and Nimbus computers). Even so, we are particularly well served by the output from publishers, and by articles at all levels in magazines such as this one. User-friendly front ends like BEEBUG's MASTER ROM have enhanced the approachability of the Acorn machines far more than any piece of software for DTI-approved rivals until, say, Microsoft windows. Those Local Education Authorities that have opted for the Archimedes will find RISC OS equally accessible in this respect. Needless to say, this approachability is very much in the spirit of what the report proposes shall be vital for young users.

The writers of the report are also fond of talking about "IT capability", which they believe consists of at least five identifiable components. First among these is familiarity with methods of communication between the machine and the user. There are certainly more light-pens, touch screens, joysticks, mice and the like for Acorn hardware than probably any other make which has either had DTI approval, or is at all widely used in education.

Again we are well served. BBC micros are as much at home handling viewdata, thanks to an inbuilt Teletext mode, and running comms packages, as any micro in the field - and a good deal more so than most major competitors. There is similarly a wealth of backup resources on topics like control and real-world applications. The BBC micro long ago surpassed the RML Nimbus in the availability of suitable software, and users have never looked back. Schools and/or departments contemplating the purchase of dedicated machines for 'real world' Design and Technology Control projects would still probably be best advised to equip themselves with model Bs (second-hand), Masters or Archimedes.

By the same token, no other 8 bit machine has fared so well where sound and music are concerned. Probably only the later generation Amiga and Archimedes can rival them even now. Moreover, when considering that section of the report which deals with sound and graphics as important media for the presentation of data [para 3.7], it is the Archimedes which springs to mind. It is quite clear about what is needed, and Acorn's 32 bit machine can definitely meet those needs.

To return to Acorn's competitors once more, several (the Mac being the latest) now include an implementation or emulation of BBC Basic. This is important. The authors of the report are at pains (at least in principle) to stress pupils' awareness and expertise in uses of IT across the curriculum. Often this will depend on subject teachers who are not computer specialists. A source of support for them (and those they teach) is material incorporating programs - for example to solve chemical equations or map archaeological finds. Many such books in fact use BBC Basic as their programming lingua franca. Illustrative routines are often written in it. Clearly where pupils and teachers have some familiarity with the real thing (possibly excepting VDU and OS commands), fuller use of these texts is possible. This does, however, assume that routines are short enough to be typed in by the user as not all such programs will run unaltered on other machines.

There is a last point of comfort if you own or use BBC or Acorn equipment. Mention is made in the interim report of the advisability of developing IT skills only to a level appropriate to the context in which it is being employed. That is to say: if pupil's language needs do not require them to know all the intricacies of a particular word processor, then these should not be taught. A glance at the sheer volume of software for Acorn machines will confirm how great a lead has been established in what the report calls 'appropriate progression'. There is, for example, nothing like the range of primary language development software available for the Nimbus as for the model B. Fortunately, much of this software - if used wisely - can engender a much more child-centred approach than the *target-led* knowledge and subject-based National Curriculum envisages. Work with computer software, like much of that produced by 4Mation for example, may also play a part in offsetting the narrowness of its conception.

The conclusion which is inescapable after this two-part survey of the curricular side of the '88 Education Act is that if you own or use Acorn equipment and have a smattering of decent software, you will be well served to fulfil its terms. Whether and how the relevant Government Departments will service either that equipment, software to run on it, or our own needs for professional development is another matter. Indeed, at the recent BETT 89 exhibition, participants from NCET-MESU put forward figures in the region of £165m per annum (excluding teacher training) as likely to be necessary to support the IT component of the National Curriculum. Until we are assured that this will be forthcoming from the DES, we will continue to be doubtful that the National Curriculum can ever be properly delivered.

Game Strategy (Part 3)

In the final part of our series on game strategy, David Spencer looks at practical tree searching, and methods for speeding up the search.

In the first two parts of this series we covered the three components needed to implement a strategy game - the *Legal Move Generator*, the *Static Evaluator* and the *Tree Search*. However, the last of these, the tree search, was looked at in a very hypothetical way, with no mention of how it might actually be implemented. We will remedy this now.

In all the examples last month, I gave the impression of a complete game tree held in memory some how. This tree could then be searched at the program's leisure. However, consider the size of a real game tree for a game of chess in mid-play. As said before, there is an average of 37 possible moves, and so for a six ply tree (a reasonable search depth), the total number of nodes in the tree is:

$37^6+37^5+37^4+37^3+37^2+37+1$

which equals 2,636,996,587. Each node must hold a board position, a score, and the pointers to its children (37 in our assumption). If each node occupied 128 bytes of memory (a realistic guess), then the entire game tree will need over 314 *Gigabytes of memory (a* Gigabyte is 1024 Megabytes).

As the above memory require-ment is somewhat more than the free memory on a model B, and probably any computer ever made, it is obviously not possible to store the entire game tree at once. Instead, we need a search routine that constructs the tree as it searches it, only holding in memory the current path actually being considered. A recursive routine to do just this could have the following structure:

```
Function Minimax(board,minmax,level)
    IF level=depth
    THEN
        =StaticEvaluate(board)
    ELSE
        Push(board)
        EvaluateLegalMoves(board)
        IF minmax
         THEN score=-big number
         ELSE score=big number
    FOR  each legal move
         Restore(board)
         newboard=Move(board)
         temp=Minimax(newboard,NOT minmax,
                     level+1)
         IF minmax
         THEN
             IF temp>score THEN score=temp
         ELSE
             IF temp<score THEN score=temp
    Pop(board)
    =score
END
```

This routine, which is written in a pseudo-code that should be easy to follow, takes three parameters. The first is the current board position, the second a flag to show whether it is a maximising (TRUE) or minimising (FALSE) level, and the third the current search level. The global variable *depth* is assumed to hold the depth of search that should be applied. The routine also calls a number of idealised functions. These are (in pseudo-code):

StaticEvaluate(board) to return the score for the board position held in *board*. (*Board* would most probably be a pointer to the board definition in memory).

Push(board) to save the given board position on some form of stack.

Pop(board) to take the most recently pushed board from the stack.

Restore(board) to load the most recently pushed board without removing it from the stack.

EvaluateLegalMoves(board) to produce a list of all the legal moves from the given board position.

Move(board) to apply the next move in the move list to the given board, returning the new board position.

The operation of the Minimax function should be fairly easy to follow. Assuming that the computer is to move, and a positive score indicates a good position for the computer, then the function would be called with:

```
    score=Minimax(current board,TRUE,0)
```

The first argument is the current board position, the second is a flag that indicates whether this is a maximising (TRUE) or minimising (FALSE) level. The final parameter is the depth of the search at this point, which will be zero on the first level.

The first action of the function is to check the current search level against the global variable *depth*, which holds the depth to which we want to search. If we have reached that depth, then it means that the board position is at a leaf of the tree, and we therefore just call the Static Evaluator and return the resulting score. Otherwise, the board position is saved on a stack, and a list of legal moves from that position generated. The best score found is then initialised. If this is a maximising level, then the score is effectively set to negative infinity, while for a minimising level it is set to plus infinity. The next effect of this is that any score will appear better than the starting value. The routine then enters a loop for each possible move. The board position is first recovered from the stack, and the move in question applied. The Minimax function is then called recursively with the new board position. The minmax flag is inverted, because it alternates on each level of the search, and the current search level is incremented by one. The score returned is compared with the current score, and if it is better, it then becomes the best so far. The sense of better depends on whether it is a minimising or maximising level. This is repeated for each move, until a final score is obtained. The original board position is pulled from the stack, and the score returned as the result.

While the above routine finds the best score we can achieve, it doesn't actually tell us which move to make! To cure this we can make a special case of the first level of the search. Instead of just keeping the score, a list of the best moves found is kept. As a new move is tried, if its score is less than the best so far it is discarded; if it is equal it is added to the list of possible moves; and if it is better a new list is started. When the search has been completed, one of the moves from the list is chosen at random and played.

APLHA-BETA PRUNING

Having conquered the problem of reducing the amount of memory needed for a tree search, we now encounter a much more serious problem - that of time. Considering the six ply search again, if we totally ignore the time taken for the actual searching, and only take account of the time needed by the static evaluator, we find that the overall time taken for the search is:

$$37^6 * t$$

where t is the time taken to evaluate one board position. If we take t to be 0.005s (200 searches a second - a realistic value for a machine code routine), then the time for the whole search is just over 12,828,632s, or put another way, about 21 weeks! Most people would get rather fed-up waiting for the computer to make its move.

Assuming that we cannot speed-up the evaluation function (well not by the amount we need), we need a method to reduce the number of positions that must be examined, while still finding the best move. This may sound like we are asking for too much, but a method does exist, and is known as *alpha-beta pruning*. Put in simple English, it basically says: "When you know you're onto a loser, don't consider it any further."

To explain alpha-beta pruning properly, consider the very simple two-level binary tree in figure 1. We will apply a standard tree search which examines positions from left to right. The first two positions looked at yield scores of 5 and -3, and as they result from a minimising move, the backed up score at node b will be -3. The routine then proceeds to look at the left-hand branch from node c, and finds a score of -4.

Therefore, and this is the important bit, we already know that if we move to node c, the player can achieve a score of -4. However, if we moved to node b, the best the player could do is achieve a score of -3. It doesn't matter how good or bad the right-hand branch from node c is; we already know that because the first level is a maximising moving to node b is better than moving to c. We even know the best score that can be obtained from node a, it is -3.

If this isn't clear, try performing the search with various scores for the last node. You will discover that whatever the score, the best score that can be achieved is -3, and the best move is a to b. If we moved instead to c, and the unknown score was greater than -4, then the player would move to the -4 position. If on the other hand the unknown score was less than -4, then the player would move to there, and we would be even worse off.

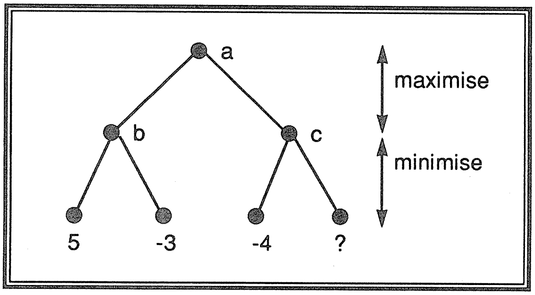


Figure 1

The net outcome is that we have performed the tree search by examining only three, instead of four, positions - a saving in evaluations (and time) of 25%. This might seem an impressive saving, but is nothing when compared with the result of applying this technique to a real game tree. For an exhaustive tree search (one with no pruning), the number of terminal nodes that must be evaluated is md, where m is the average number of moves from each position, and d is the search depth (in ply). Without explaining the theory, with alpha-beta pruning applied the number of positions that must be examined can be reduced to as little as 2*√(md). For our six ply chess game tree, applying pruning reduces the number of positions examined from over 2.5 billion to little more than 100 thousand. Looked at in terms of time, the time taken has been reduced from around five months to about eight and a half minutes. From a percentage point of view, the number of positions examined has been cut by 99.996% - quite a saving.

Notice that I have said that the number of positions examined can be as low as 2*√(md), and not that it is that figure. This is because the effectiveness of the pruning depends entirely on the order of the terminal nodes. The above expression is for the best case. In the worse case, all the positions must be examined. To illustrate this, consider figure 1 again, and let the un-evaluated score be 6. The pruning will stop the search at the -4, and as said above, only three positions are examined. Imagine, though, swapping the children of node c, as shown in figure 2. Here, the search looks at the first two nodes and obtains the backed-up score of –3 for node b just as before. However, the next position looked at has the score 6. This is better for the program than -3, and it is not until the -4 has be found that the move to node b is discovered to be better than that to node c. Therefore, we have had to consider all the terminal nodes, even with the alpha-beta pruning. In practice, the worse case is unlikely to occur, but neither is the best case.

In practice, alpha-beta pruning is achieved by using two variables called (surprisingly) alpha and beta. The value of alpha represents the best move for the opponent, and that of beta the best move for the player. As the search progresses, alpha and beta are updated as better moves are discovered. At each node, the score is compared with alpha and beta, and if it would result in the program being worse off, or the opponent better off, then that move and all moves stemming from it are rejected. More details of this can be found in reference 1, while reference 2 gives actual programming examples.

HEURISTICS

So far, all our tree searching has been based around algorithmic principles. The searching routines, whether with or without alpha-beta pruning, are guaranteed to find the highest scoring move. Furthermore, the searches have been very general. It didn't matter whether the program played chess or draughts, as long as the legal move generator knew the rules, and the static evaluator could tell a good position from bad.

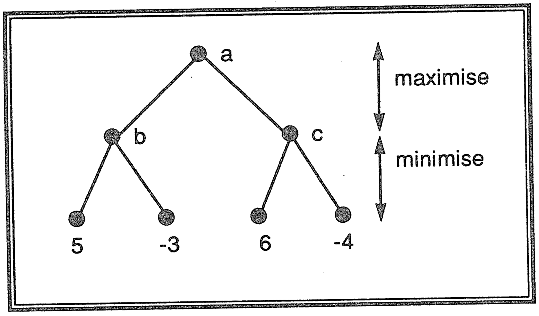


Figure 2

The next step in improving speed (and hence performance) is to apply so-called *heuristic* principles. These are methods and techniques which rather than having a firm foundation are devised from observing the playing of the game in question over as long a period as possible. Being aware of the heuristic principles of playing a game could be thought of as having a 'feel' for the game, and most people would agree that this is what makes a good player. However, unlike the techniques covered so far, heuristics can be very hit and miss. It is, for example, possible that the program will spend time examining a 'whim' or 'hunch' suggested by a built-in heuristic, only to find it leads no where. In this case, it is still necessary to perform an exhaustive search and the time spent on the heuristic is wasted.

There isn't the space to go into the details of possible heuristics, but I shall mention a couple of common ones used by chess programs. The first, and possibly most powerful is the *killer heuristic*. This relies on two facts. The first is that alpha-beta pruning is most efficient when the best moves are examined first. The second is that a move that increases a player's score by a large amount will in general be crippling for the opponent for several moves to come, and is therefore normally a good move. The killer heuristic maintains a 'best move' for the next few ply from the current position. When the tree search is performed, the 'killer' for a particular level is applied first when that level is searched. This will hopefully allow the alpha-beta pruning to be at its most efficient. The killer heuristic fails totally if a move at a higher level is such that the killer move is no longer the best move, or even no longer a valid move. In this case the search is applied normally, and a new set of killer moves generated afterwards.

THE SHANNON TYPE B STRATEGY

So far, we have performed our tree search to a fixed depth regardless to what state the game is in. However, Claude Shannon, an artificial intelligence researcher who wrote about chess playing in the late 40's, proposed a better technique which he called his type B strategy (the type A search is the fixed depth one).

Shannon's suggestion was simple. Because the game becomes more unstable when pieces are vulnerable, or are actually taken, or any similar situation, search deeper at these points. In other words, while the game is just ticking over search only to a depth of a couple of ply. However, as soon as it starts to look interesting, for example if one player loses a major piece, increase the search depth to see what transpires.

THE END GAME

We have come to the end of our look at strategy games. I don't pretend to have explained everything, or in enough depth to enable you to write your own game, but I do hope that it has given you an insight into how strategy game programs work. More details of the topics covered can be found in the two references.

References.
1. *The Chess Computer Handbook*
   by David Levy
   published by Batsford
2. *Artificial Intelligence*
   by Patrick Henry Winston
   published by Wiley and Son.

File Handling For All (Part 9)

by David Spencer and Mike Williams

This month, we move away from the topic of related records and relational databases, and turn our attention to the requirements of searching, and the subsequent organisation of records within a file.

SEARCHING

Searching is an operation which is absolutely critical to any database system. In all our examples so far, the user has not been able to search for particular records. If they had wished to look at one particular record, they would have had to scan each one manually in turn, picking out the required one when it was found. This may be OK for an example system with only a few records, but would you want to search several thousand names and addresses manually?

THE KEY FIELD

As we have seen in earlier articles, each individual record usually consists of several separate fields, for example name, address, telephone number and date of birth. When we search for a record, or a group of records, we will not normally be interested in all the fields. It is very unlikely that we would want to search for somebody called Joe Bloggs living at 34 Acacia Avenue Newtown, tel. (0123) 45678, and born on 23rd January 1956. Instead, we are far more likely to want to search for everybody in our database called Joe Bloggs, and then discover the remaining details when the records are found. Or, we might want to locate everybody whose birthday was on a particular date, regardless of who they were, where they lived etc. In fact, if we needed all the information about a record to search for it, the search would be pointless, because we already know everything held in the record we are looking for.

Therefore, when we specify a record to search for, rather than supplying all the fields for that record, we just give one or more *key* fields. The program can then search through all the records in turn, and pick out those with entries in the key fields matching those that were specified. All other fields are ignored by the search. In most cases, only one key field is specified, for example when searching for everybody with a particular name. In some cases two fields are used, such as a bank's computer that searched for people who were overdrawn and who had not been sent a nasty letter yet. It is fairly rare to encounter more than two key fields.

The idea of key fields can be taken one step further. Remember our earlier example of wanting to search for everybody born on a particular day. If the 'date of birth' field in each record held the day, month and year, then our key field scheme would only allow just to find people born on a particular day in one particular year. We need to be able to use just part of a field as the key. In our example we need to match the day and month, but ignore the year. A search routine that allows this to be done is called a *Skeletal search*.

The program can perform the search by building up a *template* record which contains the key fields to be searched for. This template has the same structure as a normal record, but only the key fields are filled in. The others, those that don't matter to the search, are either left blank, or filled with a universal *wildcard* that will match anything (rather like using a filename of '*' under DFS/ADFS). Partial fields can also be specified using wildcards. For example, if a date was stored in the form 'dd-mm-yyyy' (e.g. 23-03-1964) then to search for a particular day and month, ignoring the year, you could use a search string of 'dd-mm-*' (e.g 23-03-*). This does of course assume that all the fields are stored as strings, as we have assumed in the past.

ENTERING THE TEMPLATE

From a user's point of view, the template record for a search can be entered in much the same way as a normal record, and the same data entry routine could be used. This not only reduces the amount of program code necessary, but also ensures that the user interface is as consistent as possible throughout - an essential consideration for a user-friendly package. The user could fill in the fields on this template record, simply ignoring those which don't matter to the search, and putting wildcards in where only partial fields matter. The program can then translate this record into a template suitable for the actual search.

SEARCH AWAY

Having dealt with the creation of the template record, our next step is to perform the actual search. Assuming that we already have routines to read a record, and to print the contents of a record, the actual searching can be quite trivial. In pseudo-code, the process would be something like:

```
Move to start of file
WHILE more records
Read next record
FOR each key field
   Compare field with template
IF all key fields match
   THEN print record
ENDWHILE
```

This merely reads all the records in, one at a time, and compares the key fields of the template against those the record read in. If all the key fields match, then a match for our search has been found, and the record is printed out. Otherwise, the search continues until all the records have been read.

That is all we will say about searching for now. We will come back to the subject next month when we consider the related topic of sorting, and show how the two are beneficial to each other.

RECORD SEARCHING

As searching for a particular record, or set of records, will invariably involve reading several records in succession, the time for the overall search depends on the time taken to read records. This is of particular significance if a large number of records must be read before finding the correct one. For example, imagine an address book system with all the records searched in alphabetical order. The time taken to find Mrs. Zachrias would be a lot longer than that to find Mr. Able.

There are two ways of speeding up the search. Firstly, we could try to speed up the reading and checking of each individual record, and secondly, we could reduce the number of records that must be searched before finding the correct one. Generally speaking, the time taken to access a record is out of our control. For example, with file on disc, the time taken to read a record at random from the file will depend on the speed of data transfer from the disc, the time taken to move the drive head from one place to another, the current whereabouts of the drive head etc. While this time might vary considerably from record to record, there is very little we can do easily to reduce it in any way.

Therefore, to speed up the searching process, we have to reduce the number of records that must be examined. If our records are ordered in some way then this should be fairly straightforward, because we can predict where a record will be in relation to another one, and hence, we know where to look for a given record. In the case of totally unordered records, it might seem impossible to be able to search for any record without having to read every single record in the worst case. However, a technique called *Hashing* comes to our help. We will end this month by explaining one form of hashing, although there are many other ways of using this technique.

HASHING

Imagine the case of *n* records connected together in a list. This could be an explicit linked list, with pointers from one record to the next, or an implicit list, with successive records occupying successive positions in the file. We will now search this list for one particular record. In the best case, our record will be the first in the list, and only one record must be read from the file. In the worst case, our record will appear last, and a total of *n* records must be read to find it. If our record is equally likely to appear anywhere in list (very probable with real records), then the average numbers of records that must be read to find our is the average of the best and worse cases, numerically, (n+1)/2. This is exactly what we would expect. On average, we must search halfway through the list. Some records will be found before the halfway point, while some will be after it, but they will even out and leave the halfway average.

If we were to reduce the value of *n*, we would reduce the average number of records that must be examined, and hence the average search time. As *n* is the number of records in our list, we can only reduce it by having several short lists instead of one long one. This might sound silly, because although we will have reduced the average search time for one list, we will still have to search all the lists, because our record could be in any one of them. However, suppose we could predict in which list our record is likely to occur. Then, we would only have to search one list, and we would benefit from the shorter length lists.

All we need is a method of deciding in which particular list a certain record should occur. When records are entered we can put them in the appropriate list, and when we search for a record, we need only search the list where we would expect to find it. The process of hashing involves using the contents of a record (or more correctly an individual field) to determine which list to use. This is achieved using a so-called *hashing function*. This 'function' takes the contents of a particular field, and by some process, converts it into a list number (or a pointer to the list, etc.). So, what actual operations must be applied by the hashing function? The answer is that it doesn't really matter. As long as the function can take any valid field and convert it into any valid list number it doesn't matter how it does it. This is because we are not interested in the actual list chosen - all we require is that if the same field contents are applied to the hashing function twice or more, the list chosen is the same each time.

Does this mean that we can use any function for the hashing? No, of course not. Our whole aim to to create a number of separate lists, all of which are of a similar length. This is important, because if our hashing function puts more records into one list than another (on average), then it defeats our original purpose. This is because we are still forced in some instances to search a relatively long list. The ultimate example of an unsuitable hashing function is one which always chooses the same list, regardless of its input.

The main criterion for choosing the hashing function is therefore that all the possible list numbers will be produced with equal frequency, for a random sample of input. It is important that when choosing the function, we also take into account the distribution of our input data. Suppose, for example, we have a hashing function that takes a word as input, and produces a list number between 1 and 26. An obvious way to do this would be to choose the list according to the first letter of the word. In this way, all words beginning with 'A' go in list one, those beginning with 'B' in list 2, and so on. This will work fine if each letter of the alphabet is equally likely to occur as the first letter of a word. But flick through any dictionary - there are a lot more words beginning with, say, 'E' and 'T' than with 'X' and 'Z'. Therefore, our hashing function will result in long lists corresponding to 'popular' first letters, and shorter lists for 'unpopular' letters. How much of a problem this is depends a lot on the application in question, and the number of words being stored. With only a few words in the lists, the difference in lengths may not be a problem, but in the case of a spelling checker, where an entire dictionary is stored, the problem is much greater.

As an example of a hashing function, the following Basic function takes a character string, which could be a field from a data record, and converts it in a list number between 0 and 9:

```
DEF FNhash(field$)
field$=field$+"     "
hash=0
FOR pos=1 TO 5
hash=hash+ASC(MID$(field$,pos,1))
NEXT
=hash MOD 10
```

This function works by adding together the ASCII codes of the first five characters of the input string, and then reducing the result modulo ten to give a number between 0 and 9. The reason for using the first five characters is that this will largely remove the distribution problems caused by certain letters being more common than others. (If you think about it, there are few five letter sequences that crop up at the start of a word more often than others). The second line of the function pads out the string with spaces to ensure that it is at least five characters long. Note the use of the MOD operator to produce a result in the correct range. By changing the value of 10 in the last line, the function can be made to produce any number of list numbers. For example, if ten separate lists resulted in each list getting too long, we could change the number to 20, and have 20 individual lists.

Hashing sounds a wonderful solution, but you may have already spotted a potential problem. The hashing function only works on one particular field of a record. For example, if we had a set of records each containing a persons name and address, and we used a hashing function that converted the name into a list number, then this would be of no use if we needed to search for a particular address. This is because the address contained in a record doesn't determine which list it is in. We are again reduced to having to search every record in every list.

There are a couple of ways around this problem. Firstly, we can cut our losses and only use the hashing function on the most common key field. In our above example, although searching for a particular address required all the records to be searched, searching for an individual name could still take advantage of the hashing function. It might well be that our particular need involves searching for names much more commonly than we have to search for addresses. We can therefore live with the relatively slow address based search, as long as the name search is fast. If the situation was the other way round, in other words address searching was more common than name searching, then we would simply arrange for the hashing function to operate on the address field rather than the name field.

HASHING TABLES

There will, however, always be cases where each field in a record is equally likely to be the key field of a search. In this case, our simple hashing function is useless, and we must resort to using what are called *hashing tables*.

Imagine that all our records have a unique record number, which in the simplest case could be the record's position in the file. For each field in the record we will have a separate hashing function, each of which can produce a value between, say, 0 and 9. For each of the individual hashing functions, we will have ten lists, one for each possible value produced by the function. Instead of these lists containing records as before, they will now contain record numbers. This can be thought of as a matrix, or table, with hashing record fields on one axis, and list numbers on the other. Each entry in the matrix would be an individual list. Such as matrix is called a *hashing table*. Initially, when the database is totally empty, all the lists in the table are empty. When a new record is entered, each field is processed individually. The hashing function for that particular field is applied to find a list number, and then the record number of the new record is entered into the corresponding list. This is repeated for each field in the record.

Subsequently, when a search is being performed, the hashing function corresponding to the key field is applied to that field to obtain a list number. The appropriate list is then examined to find all possible records which could match the given key field. In the case of multiple key fields, this process could be applied for each key, and a set of lists obtained. Only records whose record number was in all of these lists are possible candidates for a match. This can greatly increase the speed of searching.

Next month we will investigate the subject of sorting.

ARCHIMEDES EXPANSION CARDS FROM WILD VISION

CHROMA 300 –

Video Overlay and Genlock System. Enables Archimedes generated graphics and text to be superimposed in full colour on a video signal from a camera or VCR.
Ideal for captioning videos.

HAWK V10 –

Image Processing System designed to match the high speed of the Archimedes. Captures, stores and displays up to 4 video images in real time independently of the host Archimedes.

AD1208 –

High Speed Analogue/Digital I/O Card 12-bit, 8 channel analogue to digital converter, 166000 samples per second and 8-bit "user port" digital I/O.

Full information on these easy to fit cards from:

Wild Vision
6 JESMOND ROAD · NEWCASTLE UPON TYNE · NE2 4PQ
TEL: 091 281 8481

EVEN THE GREATEST PERFORMANCES CAN BE IMPROVED

32K ROM **WORDWISE PLUS II** **For Model B, B+ and Master**

WORDWISE PLUS II (WW+2) is a complete word-processing system which is easy to fit and straightforward to use.

Simply remove your existing word-processor chip (if any), and plug in the WW+2 ROM. One command, and you immediately have a very powerful word-processor system.

The red function keys are used to full effect, with a keystrip showing the purpose of each one.

WW+2 is equally at home with a short letter, or a complete book.

Many operations are menu driven, so that the user is always prompted when necessary. Because it's so user friendly, it makes an ideal choice for anyone rather apprehensive about word-processing.

At the same time, WW+2 has all the features of many more expensive programs. Automatic headings and footings, page/line numbering, word counts etc. Move, copy and delete operations on text are vital in text editing, and simplicity itself.

WW+2 works with any BBC compatible printer, and that means most dot-matrix, daisy wheel and laser printers. There are no hidden extras needed in the form of special printer drivers.

What other word-processing package provides you with everything you need to do mail-shots, print information onto sticky labels, write documents of virtually unlimited length ("continuous processing") etc?

Some of the new WW+2 extensions

Many Interword-type features are present, such as menu driven loading and saving of text. Extra CTRL keys are provided to delete a line, mark a word or paragraph, search for a given string etc. Search and replace operations can now be performed on segments too. Switching between the text area and any segment requires just a single key press instead of five.

Loading and saving of text is about 10 times faster than WORDWISE. Full safety prompts make it almost impossible to lose text by accident.

New embedded commands are included which use the computer's real time clock to insert the date into a document automatically.

Naturally, existing users of WORDWISE/PLUS will find that their text files and segment programs are 100% compatible with WW+2. The (extended) programming language is much faster than WORDWISE PLUS.

Programs supplied

Provided with WW+2 are two discs of example programs. Programs on disc have the advantage that they may be changed to suit the users needs, and include a mail-merge program, label printer, and many other utilities.

A more detailed information sheet is available on request.

We have received many letters of praise from WW+2 users, and are grateful for permission to quote the following;

"..every bit as good as I was led to believe. So good, indeed, that I would recommend all WORDWISE PLUS users to upgrade to WW+2." D Sims, Cambridge.

"..excellent additions to an already outstanding piece of software."

"..delighted with all of the extra commands that WW+2 offers".

Upgrades from WORDWISE/PLUS are available.

28 day money-back guarantee!

WORDWISE PLUS II Prices (inc. VAT)

Upgrade from WORDWISE PLUS	£37.95
Upgrade from WORDWISE	£47.15
WORDWISE PLUS complete	£56.35
Spell Master (£50 when purchased with WW+2)	£51.00
16K sideways RAM module	£16.00

Please allow £1 postage

Access

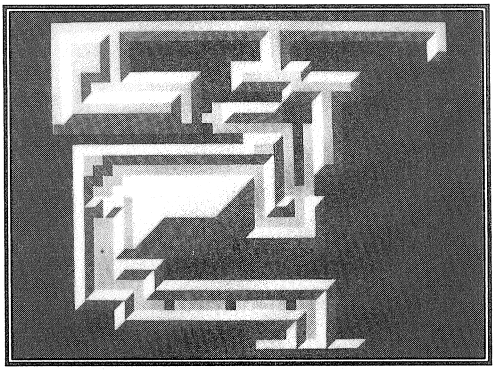
Order with Access on (0752) 847286

IFEL
36 Upland Drive
Derriford
Plymouth PL6 6BD
(0752) 847286

Roller Rally

John McFarlane's nifty programming will provide a severe challenge to your concentration, ingenuity and manual dexterity. What more could you ask for?

This game runs along the lines of *Marble Madness* with one big difference - it shows a view from above, and uses different shades of blue and white to represent a three dimensional surface. The result is impressive mode 1 graphics. You control a ball-bearing which can be moved (depending on your skill) over this surface. The idea of the game is to finish a course in as little time as possible without falling down any holes. You should move towards the bottom of the screen and as you reach it more of the course scrolls up. The game may sound easy, but it will likely prove a tantalising and frustrating test of your keyboard skills.

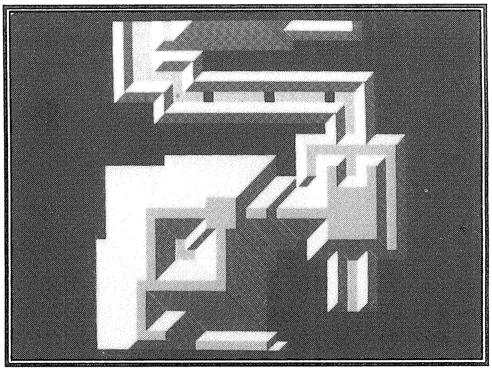

Roller Rally - screen no. 1

The program is quite short, and should be typed in and saved before playing the game. The 'Z' and 'X' keys move the ball left and right, the ':' and '/' keys move it up and down. Continuous movement in any direction enables the ball to gather momentum. When the ball is on a slope it will roll down it. To get up a slope you must either charge at it to gain sufficient momentum, or rock from slope to slope to build up enough speed. However, falling off the track and down a 'hole' will terminate your progress and return you to the start of the course.

Note that the data for the graphics and course are at the end of the program. Editing these lines will enable you to change the game if you wish. In particular, zeros in lines 2010 onwards represent holes. If these prove too difficult to negotiate, try changing them to be the same values (digits or letters) as adjoining positions. Take care too, when typing the program in, or you may find a mistake here makes the game even more difficult, or even impossible. If you use a monochrome screen and find the screen display difficult to decipher, try changing line 1020 to read:

```
1020VDU19,1,2;0;:VDU19,2,6;0;
```

A second version of this game, called Roller Coaster is included on this month's magazine disc/tape. This uses a separate data file to produce game displays rather more quickly than the scrolling version listed here.


Roller Rally - screen no. 2

NOTE: the program below is listed with a WIDTH 42 rather than our standard WIDTH 40, as many of the DATA statements are much more readable with this format.

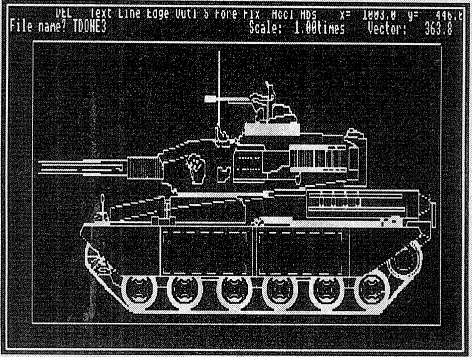
```
  10 REM Program ROLLER RALLY
  20 REM Version B1.3
  30 REM Author  J.McFarlane
  40 REM BEEBUG  March 1989
  50 REM Program subject to copyright
  60 :
 100 ON ERROR GOTO 210
 110 REPEAT
```

```
  120 MODE7:PROCintro
  130 MODE1:PROCinit
  140 PROCgamestart
  150 PROCmove
  160 PROCgameover
  170 IF Y%<32 IF A%<Hisc% PROChigh
  180 UNTIL FALSE
  190 END
  200 :
  210 MODE7
  220 IF ERR<>17 REPORT:PRINT" at line ";E
RL
  230 *FX15,0
  240 END
  250 :
 1000 DEF PROCinit
 1010 VDU23;8202;0;0;0;
 1020 VDU19,1,4;0;:VDU19,2,6;0;
 1030 VDU23,128,64,224,224,64,0;0;
 1040 RESTORE
 1050 FOR A%=&900 TO &9FF:READ?A%:NEXT
 1060 PRINTTAB(14,5)"ROLLER RALLY";TAB(13,
7)"By J.McFARLANE";TAB(13,10)"SPACE to STA
RT"
 1070 PRINTTAB(5,15)"The shortest time is
"STR$Hisc%" seconds"'TAB(15)"by "Hisc$
 1080 REPEAT:UNTIL INKEY-99
 1090 ENDPROC
 1100 :
 1110 DEF PROCgamestart
 1120 L%=0:Y%=-12:X%=639
 1130 x%=0:y%=0:Mem%=&3000
 1140 FOR Z%=0 TO 31:PROCprint:NEXT
 1150 ENDPROC
 1160 :
 1170 DEF PROCprint
 1180 IF L%=62 ENDPROC
 1190 VDU4:PRINTTAB(0,31):READ A$
 1200 FOR A%=1 TO LENA$:C%=EVAL("&"+MID$(A
$,A%,1)):IF C%=0 GOTO1220
 1210 FOR B%=0 TO 15 STEP4:B%!(Mem%+A%*16-
16)=B%!(&900+C%*16):NEXT
 1220 NEXT:Mem%=Mem%+640
 1230 IF Mem%=&8000 Mem%=&3000
 1240 Y%=Y%+32:L%=L%+1:VDU5
 1250 ENDPROC
 1260 :
 1270 DEF PROCmove
 1280 TIME=0:VDU5,18,4,0
 1290 REPEAT:MOVEX%-4,Y%+4:VDU128
 1300 IF Y%<160 FOR Z%=0 TO 19:PROCprint:N
EXT
 1310 FOR A%=1 TO 100:NEXT
 1320 y%=y%+(INKEY-105-INKEY-73)
 1330 x%=x%+(INKEY-98-INKEY-67)
 1340 A%=POINT(X%,Y%):B%=POINT(X%,Y%-4)
 1350 IF A%+B%=1 y%=y%+3
 1360 IF A%=0 AND B%=0 x%=x%-3
 1370 IF A%+B%=5 x%=x%+3
 1380 IF A%=2 AND B%=2 y%=y%-3
 1390 IF TIME MOD8=1 y%=y%-SGNy%:x%=x%-SGN
x%
 1400 IF ABSy%>11 y%=11*SGNy%
 1410 IF ABSx%>11 x%=11*SGNx%
 1420 MOVEX%-4,Y%+4:VDU128
 1430 X%=X%+x%:Y%=Y%+y%
 1440 UNTILA%+B%=6 ORA%=-1 ORY%<32
 1450 A%=TIME:VDU4:CLS
 1460 ENDPROC
 1470 :
 1480 DEF PROCgameover
 1490 IF Y%>32 FOR B%=101 TO 53 STEP-4:SOU
ND1,-10,B%,1:NEXT ELSE FOR B%=53 TO 101STE
P4:SOUND1,-10,B%,1:NEXT
 1500 PRINTTAB(17,3)"TIME:";TAB(15,5)STR$(
A%DIV100+1)" SECONDS"
 1510 *FX15
 1520 A%=INKEY(500):CLS
 1530 ENDPROC
 1540 :
 1550 DEF PROCintro
 1560 VDU23;8202;0;0;0;
 1570 FOR A%=0 TO 1:PRINTTAB(13,A%)CHR$141
"ROLLER RALLY":NEXT
 1580 PRINTTAB(13,A%)"By J.McFarlane"
 1590 PRINT'"   Using Z, X, * and ? keys g
uide the"
 1600 PRINT"ball-bearing over the 3D surfa
ces and"
 1610 PRINT"down to the very bottom of the
 track."
 1620 PRINT'"   You must avoid falling int
o black"
 1630 PRINT"holes and off the edges of the
 path."
 1640 PRINT'"   If you want to go up a slo
pe you"
 1650 PRINT"must gather momentum. This can
 be"
 1660 PRINT"done in one of two ways. First
ly you"
 1670 PRINT"can take a run up if there is
enough"
 1680 PRINT"space otherwise you have to ro
ck back"
 1690 PRINT"and forth on an opposite slope
 until"
 1700 PRINT"you have picked up enough spee
d."
 1710 PRINT''TAB(14)"Press SPACE"
 1720 REPEAT:UNTIL INKEY-99
 1730 Hisc%=150:Hisc$="JOHN"
 1740 ENDPROC
 1750 :
 1760 DEF PROChigh:CLS
 1770 PRINT'TAB(12)"CONGRATULATIONS!"
 1780 PRINT'"You have the fastest finishin
g time yet!"
 1790 PRINT"Please enter your name"
 1800 INPUT Hisc$
 1810 Hisc%=A%
 1820 ENDPROC
 1830 :
 1840 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 1850 DATA 255,255,255,255,255,255,255,255
,255,255,255,255,255,255,255,255
 1860 DATA 245,250,253,254,255,255,255,255
,245,250,245,250,245,250,253,254
 1870 DATA 245,250,245,250,245,234,197,138
,245,234,197,138,5,10,5,10
 1880 DATA 255,255,255,255,255,239,207,143
,255,239,207,143,15,15,15,15
 1890 DATA 5,10,13,14,15,15,15,15,5,10,5,1
0,5,10,13,14
 1900 DATA 7,11,5,10,5,10,5,10,15,15,15,15
,7,11,5,10
 1910 DATA 15,15,15,15,15,31,63,127,15,31,
63,127,255,255,255,255
 1920 DATA 5,10,5,10,5,26,53,122,5,26,53,1
22,245,250,245,250
 1930 DATA 247,251,245,250,245,250,245,250
,255,255,255,255,247,251,245,250
 1940 DATA 245,250,245,250,245,250,245,250
,245,250,245,250,245,250,245,250
 1950 DATA 5,10,5,10,5,10,5,10,5,10,5,10,5
,10,5,10
 1960 DATA 15,15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15
 1970 DATA 240,240,240,240,240,240,240,240
,240,240,240,240,240,240,240,240
 1980 DATA 245,250,253,254,255,239,207,143
,245,234,197,138,5,10,13,14
 1990 DATA 7,11,5,10,5,26,53,122,15,31,63,
127,247,251,245,250
 2000 :
 2010 DATA DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDD
 2020 DATA D2AAA3D6CCCCCCCCCCCCC7D6CCCCCCC
CCCCCCC7D
 2030 DATA D1DDDBDB00000000000001DB0000000
00000001D
 2040 DATA D1DDDBDB00000000000001DB0000000
00000004C
 2050 DATA D1DDDBDB00000000002AA9D8A3
 2060 DATA D1D6C5D8AAAAAA30001DDDDDD8AAAA3
 2070 DATA D1DBDDDDDDDDDDDB0004CCCC7DDDDDB
 2080 DATA D1D8AAAAAAAA3DB000000004CC7D6C5
 2090 DATA D1DDDDDDDDDDDBDB0D2AAAA30001DB
 2100 DATA D4CCCCCCCCCC5DBDD4CCCC50001DB
 2110 DATA CCCCCCCCCCCCCC50DDDDDDDD0D1DB
 2120 DATA 00000000000000000002AA3D0D1DB
 2130 DATA 002AAAAAAAAAAAAAAAA9DDBD0D1DB
 2140 DATA 001DD6CCCCCCCCCCCCCCC7BD0D1DB
 2150 DATA 001D65DDDDDDDDDDDDDDD1BD0D4C5
 2160 DATA 00165DD2AAAAAAAAAAA3D1BD0D
 2170 DATA 001BDD292AAAAAAAAA3BD1BDDD
 2180 DATA 0018329D12AAAAAAA3BBD183D23
 2190 DATA 0016547D112AAAAA3BBBD4C5D1B
 2200 DATA 001BDD1D114CCCCC5BBBDDDDD1B
 2210 DATA 001BDD1D14CCCCCCC5B8AAAAA9B
 2220 DATA 001BDD1D4CCCCCCCCC56CCCCCC5
 2230 DATA 001BDD4CCCCCCCCCCCC5
 2240 DATA 001BDD2AAA3
 2250 DATA 001BDD4CC7B
 2260 DATA 001BDDDDD18AAAAAAAAAAAAAAAA3
 2270 DATA 0018AAA3D4CCCCCCCCCCCCCCCC7B
 2280 DATA 004CCC7BDDD0DDDDD0DDDDD0DD1B
 2290 DATA 00000018AAAAAAAAAAAAAAAA3D1B
 2300 DATA 0000004CCCCCCCCCCCCCCCC7BD1B
 2310 DATA 000000000000000000000001BD1B
 2320 DATA 000000000000000000002AA98A98AA3
 2330 DATA 000000000000000000001DDDDDDDDDB
 2340 DATA 00000002AAAAAAAAA3001D2A3D2A3DB
 2350 DATA 02AAAAA92AAAAAAA3B001D1DBD1DBDB
 2360 DATA 012AAAAA92AAAAA3B8A3471D8A9DBDB
 2370 DATA 0112AAAAA92AAA3B8A38A91DDDDDBDB
 2380 DATA 01112AAAAA9DDDB8A38AAA9DDDDDBDB
 2390 DATA 01111DDDDDDDDDBDDBDDDDDDDDDDBDB
 2400 DATA 01111D6CCC7DDDB6C56CCC7DDDDDBDB
 2410 DATA 01111DB6C747D65B6C56C71DDDDDBDB
 2420 DATA 29111DBBD471DB65B6C5014CCCCC5DB
 2430 DATA 12911DBBDD11DBB65B0004C76C76CC5
 2440 DATA 11291DB8AA91DBBB65000001BD1B
 2450 DATA 11129D8AAAA9DBBBB0000001BD1B
 2460 DATA 1111DDDDDDDDDBBBB0000001BD1B
 2470 DATA 1111D6CCCCCCC5BBB0000001BD1B
 2480 DATA 1111DB6CCCCCCC5BB00000045D45
 2490 DATA 1111DB8A36CCCCC5B
 2500 DATA 1111D8A3BB6CCCCC5
 2510 DATA 1111DDDBBB8AAAAAA3
 2520 DATA 1114CCC5BBDDDDDDD83
 2530 DATA 114CCCCC5B6CCC76C7B
 2540 DATA 14CCCCCCC5B0001BD1B
 2550 DATA 4CCCCCCCCC50001BD1B
 2560 DATA 000000000000001BD1B
 2570 DATA 000000000000001BD1B
 2580 DATA 000000000000001BD1B
 2590 DATA 000000000000001BD1B
 2600 DATA AAAAAAAAAAAAAA9BD18AAAAAAAAAAAA
AAAAAAAAA
 2610 DATA CCCCCCCCCCCCCC5D4CCCCCCCCCCCCCC
CCCCCCCCC
 2620 DATA DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDD
```

ASTAAD

The Best of BEEBUG

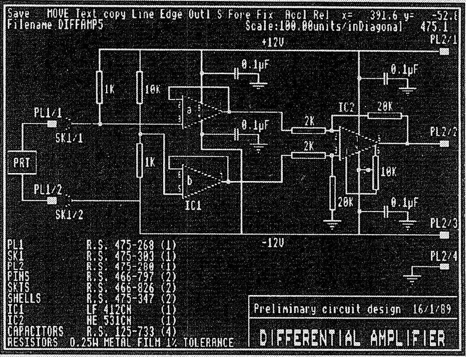
Our new BEST OF BEEBUG disc is based on the recent series of articles and programs dealing with ASTAAD, our comprehensive CAD program for the Master and Compact. The ASTAAD disc contains everything you need to use this very impressive piece of software, including the enhanced versions of the ASTAAD and STEAMS programs. This is supplied complete with documentation and printed keystrips for use with ASTAAD and STEAMS.



```
DEL   Text Line Edge Outl S Fore Fix  Accl Abs    x=  1003.0  y=  446.
File name? TDONE3                         Scale:  1.00times    Vector:   363.8
```

ASTAAD £9.95

- * Enhanced ASTAAD CAD program now including full mouse and joystick control, built-in printer dump and speed improvement.
- * STEAMS image manipulator
- * 8 page manual covering the use of the system.
- * Keystrips for ASTAAD and STEAMS
- * Sample picture files to experiment with.



```
Save   MOVE Text copy Line Edge Outl S Fore Fix  Accl Rel   x=   391.6 y=   -52.
Filename DIFFAMP5                            Scale:100.00units/inDiagonal   475.1
```

```
PL1          R.S. 475-268 (1)
SK1          R.S. 475-303 (1)
PL2          R.S. 475-280 (1)
PINS         R.S. 466-797 (2)
SKTS         R.S. 466-826 (2)
SHELLS       R.S. 475-347 (2)
IC1          LF 412CN     (1)
IC2          NE 531CN     (1)
CAPACITORS   R.S. 125-733 (4)
RESISTORS  0.25W METAL FILM 1% TOLERANCE
```

Preliminary circuit design 16/1/89

DIFFERENTIAL AMPLIFIER

Recent magazines have shown many examples of the truly impressive results that can be achieved with ASTAAD for producing technical drawings and diagrams including perspective work. The disc also contains several example drawings for you to load and modify, for easy familiarisation with the software. All this is at a fraction of the price that you would need to pay for equivalent commercial software.

Our first two releases under the Best of BEEBUG label are still available:

General Utilities £5.75
Best of BEEBUG Disc 1

* Printer Buffer * * ROM Controller * Sprite Editor/Animator * Multi-Character Printer Driver for View * Mode 7 Screen Editor * Multi-Column Printing * Epson Character Definer * BEEBUG MiniWimp † * ROM Filing System Generator

*** Master series only. † Requires sideways RAM.**

Applications £5.75
Best of BEEBUG Disc 2

* Business Graphics * Video Cataloguer * World by Night and Day * Phone Book * Page Designer * Personalised Letter-Heads * Mapping the British Isles * Selective Breeding * Appointments Diary * The Earth from Space * Personalised Address Book

Please rush me my Best of BEEBUG discs:

ASTAAD Code 1407A *(80 track DFS)* ☐
Code 1408A *(3.5" ADFS)* ☐

General Utilities Disc Code 1405A ☐
Applications Disc Code 1404A ☐

Name ______________________

Address ______________________

____

Memership No ______________________

Total £________

Postage (£0.60 + £0.30 for every other disc) £________

Grand Total £________

I enclose a cheque for £__________ OR please debit my Access, Visa or Connect account, Card No_______/_______/_______/_______ Expiry_____/_____ Signed__________________

Return to BEEBUG Ltd, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX. Telephone (0727) 40303.

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

This month's hints and tips have been collected together by Mike Williams. Remember we pay five pounds for each hint published and fifteen pounds for the star hint.

* * * STAR HINT * * *

CLOSING FILES BETWEEN PROGRAMS

John Collyer

If the same data file is to be used in several programs in the same suite, there is no need close the file between programs. However, you should use one of the global integer variables A%-Z% to record the channel number so that you can access the file in each program.

WORDWRAP

Al Harwood

To keep the output of large quantities of text to the screen tidy, use the following procedure to provide word wrap, that is to avoid splitting words between lines, where w$ is the whole of the text message (limited to 256 characters).

```
 1000 DEF PROCwrite(w$)
 1010 LOCAL A,P,L,LL,A$
 1020 LL=40:P=1
 1030 REPEAT
  1040 L=LENw$-P:FOR A=P TO LENw$:IF MID$
(w$,A,1)=CHR$32 L=A-P:A=LENw$
 1050 NEXT:L=L+1:IF L+POS>LL PRINT
 1060 A$=MID$(w$,P,L)
  1070 FOR A =1 TO LENA$:PRINT MID$(A$,A,
1);
 1080 NEXT:P=P+L:UNTIL P>LENw$
 1090 ENDPROC
```

The procedure does not output a Carriage Return at the end of the message, but this could be added by appending ':PRINT' to the end of line 1080. At the moment the procedure is set up for forty column screens, but this can be changed in line 1020.

TO CLOSE OR NOT TO CLOSE

Sheridan Williams

On occasions where several files are open at once, you may want to close an individual file (on channel c say) without knowing whether that particular file was open or not. CLOSE#c would not do, because if the file hadn't previously been opened, c would have the value 0, and all files would be closed as CLOSE#c would equate to CLOSE#0. You could use the following to close the file on channel c:

```
        IF c>0 CLOSE#c:c=0
```

but this is tedious if it is to be used in many places in your program. The following function achieves the same effect:

```
        DEF FNclose(x)
        IF x>0 CLOSE#x
        =0
```

To close a file on channel c (say) you would then use:

```
        c=FNclose(c)
```

CHARACTER DEFINITIONS

Donald Tattersfield

The method of creating your own graphics characters using the VDU23 command is well described in the User Guide. The row sums may be given in decimal or hex. Surprisingly, combining the two can be easier and quicker. Bisect the 8x8 grid as shown in figure 1. For each row sum separately the left and right hand parts and combine as shown. The example character definition becomes:

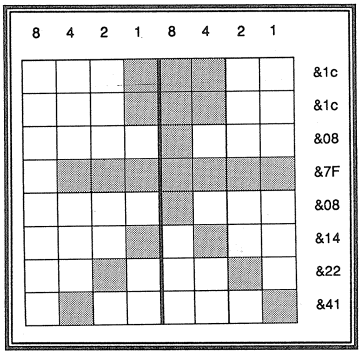


Figure 1

```
VDU23,240,&1C,&1C,&08,&7F,&08,&14,&22,&41
```

MODERN ART

V.W.Patterson

For a continuous tableaux of 'modern art' on a Master try the following one-liner:

```
MODE129:CLS:REPEAT:GCOL0,RND(7):MOVE RND
(512),RND(640):VDU25,RND(207),RND(512);RN
D(640);:UNTIL FALSE
```

First Course: The Ins and Outs of Basic (continued from page 41)

separately, whereas the more normal INKEY returns an ASCII code telling us which key has been pressed and therefore only has to be tested once.

Although most negative INKEY functions are listed in the User Guides under the keyword, one is often missing - INKEY-256. This returns a different value depending on which system is being used - 255 for a model B (1.0 & 1.2 OS), 253 for a Master 128, 245 for a Compact and so on.

Do remember that negative INKEY looks directly at the keys, and is quite independent of the contents of the keyboard buffer. Likewise, flushing the keyboard buffer is quite irrelevant as far as negative INKEY is concerned.

MAKING AN EXIT

We will finish this section with a simple but effective use of the negative INKEY function. Although a good many programs use Escape to exit from the program, as many more use Escape to terminate the current action and return to a menu or similar. The problem in the latter case is how to arrange an effective exit. IF a menu is being used, then putting Exit as one of the menu options may be the solution.

The other alternative is to use Shift-Escape to exit. Assuming that the pressing of Escape is handled by an error trapping routine (what follows can still be adapted for other circumstances), then the coding would look something like this:

```
   IF ERR<>17 THEN REPORT:PRINT" at line "
;ERL:END
   IF INKEY-1 THEN MODE7:END
   GOTO 150
```

where INKEY-1 detects the Shift key. If some other error has occurred this is reported in the usual way and the program terminates. If Escape has been pressed, negative INKEY is used to check if Shift was pressed at the same time, and if so the program terminates. Otherwise a jump is made to a suitable point at which the program can continue, usually the start of the main program loop.

That concludes our discussion of input and output in Basic. I am sure there is probably much more that I could have said (let me know if there is), but I hope that in these three articles I have given you some useful ideas, and left you with plenty to think about.

Colossus Bridge 4 (continued from page 35)

making sensible decisions on most occasions. Opening leads are defined in the manual, but to give you a flavour this is what it says "Partner's bid suit, top of a 3 card honour sequence, Ace from AK, King from KQ, 4th highest of longest suit (no trumps), singleton or high-low from a doubleton. If sensible, partner's lead will be returned."

♥ THE TUTOR

This contains a series of ten hands illustrating the fundamental principles of the game. On each hand the player must enter the correct pre-determined bid and then play the hand before seeing a description and analysis. The tutor is disappointingly in black and white, and it would be nice to see the hand again after the analysis. Hand 6 gets N/S and E/W confused in the analysis. I found overall that I could have done with more than 10 hands. Treat the Tutor as a bonus and you won't be disappointed.

♥ CONCLUSION

If, like me, the opportunities for playing Bridge are insufficient to meet your craving, Colossus Bridge will at least give you a game. During play however, it cannot meet anything other than the immediate problem. For instance it cannot anticipate impending problems such as the lead being in the wrong hand, end-play, etc. It plays according to the rules in the manual, so at least you know where you are. I found it great fun, and have been back and used it many times. For its price I cannot do anything other than recommend it, especially if you are a beginner.

```
 3460 IF Y%<16 Y%=16
 3470 PROCdrawv
 3480 UNTIL INKEY-106 OR esc%
 3490 IF esc% GOTO 3550
 3500 GCOL0,0:PROCdrawv
 3510 REPEAT:UNTIL NOT INKEY-106
 3520 xpos%=(X%-32)/4:len%=L%/8:ypos%=68
-len%-(Y%-16)/8
 3530 PRINTTAB(21,2+t%);FNp(xpos%);" ";F
Np(ypos%);"  1 ";FNp(len%)
 3540 PROClinedata(mem%):count%=count%+1
:mem%=mem%+3
 3550 UNTIL esc%:PROCdrawv:?vert%=count%
-1
 3560 ENDPROC
 3570 :
 3580 DEF PROCdrawh MOVE X%,Y%
 3590 DRAW X%+L%-1,Y%:ENDPROC
 3600 DEF PROCdrawv MOVE X%,Y%+4
 3610 DRAW X%,Y%+L%-1
 3620 ENDPROC
 3630 :
 3640 DEF PROClinedata(mem%)
 3650 mem%?0=xpos%:mem%?1=ypos%
 3660 mem%?2=len%
 3670 ENDPROC
 3680 :
 3690 DEF FNok
 3700 PRINT" Is this OK ?":*FX15,1
 3710 REPEAT G%=GET AND &DF
 3720 UNTIL G%=ASC"Y" OR G%=ASC"N"
 3730 =(G%=ASC"Y")
 3740 :
 3750 DEF FNp(P%)
 3760 =CHR$(-32*(P%<10))+STR$P%
 3770 :
 3772 DEF FNexists(F$)
 3774 F%=OPENIN(F$):CLOSE#0:=(F%<>0)
 3776 :
 3780 DEF PROCchain
 3790 PROCw2:CLS
 3800 PRINT'" Press SPACE to chain"'" th
e printer driver."
 3810 REPEAT UNTIL GET=32
 3820 CHAIN"DRIVER"
 3830 ENDPROC
 3840 :
 3850 DEF PROCw1
 3860 VDU28,1,13,38,3:ENDPROC
 3870 :
 3880 DEF PROCw2
 3890 VDU 28,14,26,38,15
 3900 ENDPROC
 3910 :
 3920 DEF FNyposn
 3930 IF t%<11 PRINTTAB(0,t%);:=t%
 3940 PRINTTAB(0,10);CHR$10;:=10
```

Points Arising....Points Arising....Points Arising....Points Arising....

ASTROLOGICAL BIRTH CHARTS (Vol.7 No.3)

In certain cases where the Ascendant falls near the cusp of two houses, it is placed in the wrong house. This is corrected by adding the line:

```
1005 IF HA>2*PI THEN HA=HA-2*PI
```

MOTORISTS ROUTE PLANNER (Vol.7 No.6)

A problem in this program causes the elapsed time to be printed as a negative value. This is cured by changing the following two lines:

```
2680 IF m%>59 THEN m%=m%-60:h%=h%+1
2690 IF h%>23 THEN h%=h%-24
```

ADFS DIRECTORY DELETER (Vol.7 No.8)

If the name of the directory being deleted is ten characters long, then it is printed incorrectly when the program asks for confirmation. This is cured by changing the following lines:

```
140 osrdch=&FFE0:length=75:REM Do NOT
increase size of length
2560 INX:CPX #14:BEQ rts
2565 CMP #32:BNE fname
2570 .rts RTS
```

GRAPHIC DESIGN WITH ASTAAD 3 (Vol.7 No.7)

An error in this program can cause text printed using the ASTAAD feature to appear incorrectly in some circumstances. This is due to a mistake in line 5090, which should read:

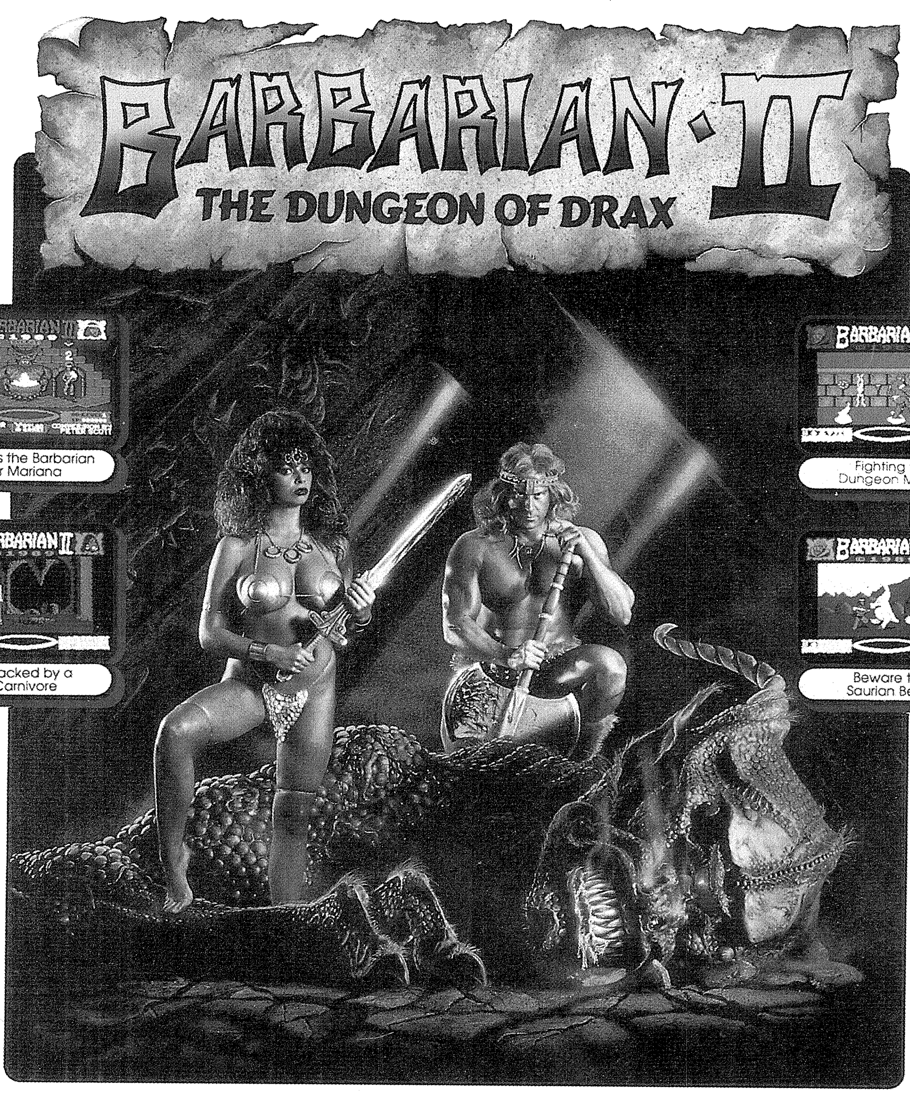
```
5090 diag=SQR(asize*asize+aheight*ahei
ght)+dotr*ratio
```

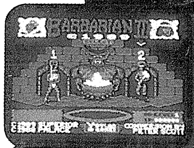
The second '*' was listed as a '+' in the original.

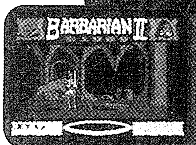
MONTHLY DESK DIARY (Vol.7 No.7)

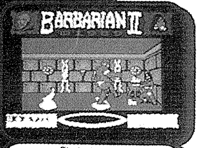
As published the program will not allow you to print out a month's data if your file contains records in a previous month. To cure this amend the following line:

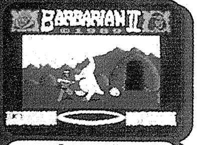
```
2090 IF inserts THEN X=OPENUP"W.DATES":
REPEAT PROCgetdaterec:UNTIL rmonth%>=
month%
```




Play as the Barbarian or Mariana


Attacked by a Carnivore


Fighting the Dungeon Master


Beware the Saurian Beast

BARBARIAN II - The Barbarian and Mariana Fight the Monsters

At the finale of BARBARIAN - THE ULTIMATE WARRIOR, the Barbarian defeated the warriors of Drax and thus freed Princess Mariana from his evil spell. Drax fled to the dungeons beneath his black castle, vowing to wreak disaster on the Jewelled Kingdom.

There is only one way to stop Drax. The Barbarian and Mariana - herself an accomplished swordswoman - are the only two warriors skilled enough to survive the perilous journey to Drax's lair. They must stop him before it is too late.

Playing the role of either the Barbarian or Mariana, you must fight your way through three levels - the Wastelands, the Caverns and the Dungeons - each being a maze of about 26 screens.

There are 17 different types of monster to defeat, including: Stabbers, Stingers, Pit Things, and Gobblers. You must also collect six different magical objects in order to survive the quest.

Finally you reach the fourth level, the Inner Sanctum, where you face the Living Idol, the Demon and finally the dreaded Drax!

BARBARIAN II - The Dungeon of Drax
© Palace Software, 1988.
Made under licence.

(The screen pictures show the BBC Micro version of the game.)

BBC Micro Cassette £9.95	Acorn Electron Cassette £9.95
BBC Micro 5¼" Disc £11.95	BBC Master Compact 3½" Disc £14.95

(Compatible with the BBC B, B+ and Master Series computers)
Please make cheques payable to "Superior Software Ltd".

OUR GUARANTEE
- All mail orders are despatched within 24 hours by first-class post.
- Postage and packing is free.
- Faulty cassettes and discs will be replaced immediately.
(This does not affect your statutory rights).

SUPERIOR SOFTWARE Limited

ACORNSOFT

PALACE

(Acornsoft is a registered trademark of Acorn Computers Ltd. Superior Software Ltd is a registered user.)
Dept AR1, Regent House, Skinner Lane, Leeds LS7 1AX. Telephone:(0532) 459453

Available from WHSMITH and all major dealers

VISA

24 HOUR TELEPHONE ANSWERING SERVICE FOR ORDERS

POSTBAG

POSTBAG

MORE SYMMETRY

A useful addition to the program for those hypnotised by Symmetrical Patterns (Vol.7 No.7) is to add lines to change modes and increase the variety of patterns. Mode 0 or 128 makes an interesting alternative:

```
  65 Z%=0
 355 IF Z%=4 THEN Z%=0:GOTO 357
 356 IF Z%=0 THEN Z%=4
 357 MODE Z%:VDU23,1,0;0;0;0;
```

Rob Barnes

GETTING TIGHT ON MEMORY

I wonder whether you can offer an explanation for the following. On several occasions when I have typed in a listing which is in mode 2, I get a "Bad Mode" message. Invariably changing the program to run in mode 5 enables the program to work. Since the listings are for a model B, surely they should run in mode 2 as stated.

A recent example did give a warning that it would be tight on memory, and that disc users should download the program. I am not very computer literate, but I tried using a downloading program without much success. Any helpful advice you can give will be much appreciated.

J.A.Meitiner

Mr Meitiner has raised a topic that confuses many people. First of all if we publish a program designed to run in mode 2 (or any other mode) then it will run in that mode on a model B unless the magazine states otherwise. What can happen is that our program listings are produced with a LISTO1 setting for readability. This prints an extra space between the line number and the start of the program proper. This space should be omitted when typing the program in from the magazine. In more extreme cases the extra spaces can make the difference between a program working correctly or not. This can apply to both disc and tape systems, but the remaining comments apply to model B disc systems only (the Master series defaults to a PAGE setting of &E00).

Secondly, if a program is still short of space, the simplest solution is to type:

```
PAGE=&1200 <Return>
```

before loading the program. This lowers to &1200 the point at which a program is loaded, therefore giving it more space.

A movedown routine (see below) is only really necessary in extreme cases where a program needs all possible memory on a model B. Add the following lines to the start of the program to be downloaded (line 1 assumes the first line of the program proper is 10), re-save, and then load and run as normal. Unfortunately, such a movedown routine cannot be used where a program needs to read or write to disc. After the program has terminated, you will need to reset PAGE to at least &1200 before you can sensibly access a disc.

```
   1IF PAGE<&E01 THEN 10
   2*K.0  *T.|MFORA%=0TO(TOP-PAGE)STEP4:A%
!&E00=A%!PAGE:NEXT|MPAGE=&E00|MOLD|MRUN|M
   3*FX138,0,128
   4END
```

BREAK CHANCING

With reference to Dr.R.Parish in Hints & Tips Vol.7 No.7, if you accidentally press Break in View, losing your text, just type:

```
OLD <Return>
```

and your text is recovered.

W.Johnsen

Unfortunately, life is not quite so simple, and both Mr Johnsen and Dr. Parish are correct. There have been at least three separate versions of View (numbered 1.4, 2.1 and 3.0) and there are the versions (effectively version 3.0) supplied in ROM for the Master, and in ROM image format for the Compact.

On the Master, pressing Break causes no damage at all, merely leaving the user in command mode rather than edit mode. This version doesn't recognise 'OLD' anyway, and nor does version 2.1. Version 1.4 does recognise the OLD command, and this can be used to recover text with that version.

With version 2.1 in particular, the system described by Dr. Parish can be invaluable, saving on some occasions many hours of retyping. The moral is, keep saving your text at frequent intervals. This not only safeguards against the accidental use of Break, but power cuts, disc failures and other nasties.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX. The normal copy date for receipt of all ads will be the 15th of each month.

WANTED: ATPL ROM board for BBC B and instructions. Tel. (0823) 289378.

Turbo 65C102 Co processor complete with Hi BASIC/Hi-Editor/Printer buffer on disc £55. SWR cartridges for Master £10 each. Care Electronics User Port Expander £20 for either BBC B or Master. Tel. (0352) 2473 after 6pm.

BBC issue 7 little used, still boxed £200 o.n.o. Tel. (0252) 702632.

Star Gemini 10X Printer, all accessories boxed £75. Tel. (0908) 679349.

Brother HR5 printer incl. power supply , cable and 6 ribbons, recently serviced £60. Watford Dumpout 3 ROM unused £15. Slogger BEEBMAN ROM unused £10. ACP 1770 E00 DFS disc and manual £5. BBC Electron cassettes included. Revs/Revs4Tracks/Aviator. Wanted Zalaga for BBC. Also Electron User mags Oct 83-Mar 88 inclusive. Offers? Tel. (0322) 68534.

Challenger 3" disc drive & 512k. Silicon disc £150. Twin 36 Voltmace joysticks £10. InterWord ROM & manuals £25. Various tape games £3 each, Wizador, 737 Flight Simulator & many more. Tel. (0993) 776066.

Sanyo 12" green screen monitor £30. Tel. 01-942 1766.

Archimedes Acorn Fortran 77 £50, Clares Graphic Writer £15. BBC AMX Page Maker £20, Realtime Solids Modeller (7 discs) £45. BEEBUG "C" system £33. Tel. 01-863 6641 (answerphone).

ISO Pascal for BBC B. Boxed unused £40. Tel. 01-650 2205.

View Professional two discs (3.5in and 5.25in), two ROMs, manual and all paper work, in original package £55. Tel. (0322) 64761.

Superior Collection Vol.1, Repton3, Around the world in 40 screens, Speech, Hunchback, The Adventure Creator £30. Tel. (0734) 441385.

BBC B OS 1.2, Watford DFS, Watford CD400 twin 40T drive, Watford Sideways ROM board with RAM chips fitted, Watford 32k shadow RAM/ Buffer, manuals leads etc. £300. Tel. (0296) 28631.

WANTED: Sheet Feeder for Epson LX80, will consider buying an LX80 and sheet feeder. For Sale: ATS ROM. £3. Printmaster ROM & manual. £15. Tel. (09295) 51450 (or 3670 eves.).

BEEBUG Toolkit ROM £7, ViewSheet ROM £18, GXR (for 'B') £15, Watford Dumpout 3 £12.50. All with manuals etc. Also "Elite" disc (for 'B') £5, "Castle Quest" disc £3. Tel. (028 488) 545.

Aries B32 & B12 shadow RAM, SWR/ ROM boards. B12 fitted with 2 6264 RAM chips £85 o.n.o. Will accept ATPL Sideways board in p/x. ViewStore package £25. Prism modem 2000 with BBC lead and ROM £25. All in good condition, complete and original. Open to reasonable offers. Tel. 051 647 5367.

Micronet ROM and Prism 1000 Modem £30. BASIC 1 ROM and Wordwise ROM, offers. Tel. (0732) 364121.

BBC Model B issue, 1770 DFS, ADFS 40/80 double sided drive, tape deck ATPL ROM board + sideways RAM many discs, tapes, ROMs, books and magazines. £350 o.n.o. Tel. (0322) 63498 after 7.30 pm.

BBC Master 128, Torch twin disc drive, Microvitec Hi-Res monitor, Teletext adaptor, Apollo Modem, EpROM programmer, large amount of software; BCPL Pascal, Forth, etc. Offers. Tel. 01-907 2283 evenings.

Aries B-32 (32k shadow/sideways RAM board) £40.InterWord £30, Viewsheet £25, BEEBUG'S Toolkit £10. Advanced user guide £8. Tel. 01-882 3552.

Electromusic Reasearch 'Scorewriter' comprising ROM, utilities disc and manual, compatible BBC B, B+ and Master. Offers over £50. Tel. (03477) 553.

BEEBUG magazines vols. 2-7, many Acorn User magazines and some programming books, including BBC BASIC in 30 hours. Offers! Tel. (02216) 2965.

BBC B issue 7 OS 1.2. Watford DDFS and Pace 40/80 DSDD disc drive. Watford 32k RAM, APTL ROM/ RAM board, Nidd Valley Digimouse and Paintbox software, CC disc doctor ROM, ROM Spell Transfer ROM, Dumpout 3, CC Graphics, Advanced Disc Toolkit, Advanced Disc Investigator, Care Master Cartridge, Latest issue InterBase, Wide selection of back issue magazines and discs from BEEBUG, Micro User, Acorn User. Will split, any sensible offers. Tel. 01-505 2931 day, or (0279) 70362 after 6pm.

BBC B expansion board for 5 ROMs £5. Dumpout 3 ROM + manual, boxed as new £10. BASIC IROM £2. TFS ROM + teletext manual £3. Tel. (06285) 30724.

Business Ad

SPROG SQUASHER

Reduce size and increase speed of Wordwise-Plus segment programs. On 5.25 inch disc/ cassette for BBC Micro. Only £5 (incl p&p). Cheque/PO payable to C. G. Robbins.

Send for details or order to
Round Robin,
29 Sandpit Lane,St. Albans,
Herts, AL1 4EW

WANTED: Viglen console (key-board unit only) for BBC B. Tel. (0785) 713855 after 6pm.

Expanded Master 512 (STL 1meg PC+), two 80T disc drives, GEM programs and mouse, Reference manuals 1&2, Disc User 1-6 and selected software, £650 or offers. Software for Electron, Beeb, Master and Archimedes from £1.50. Tel. 061 626 9170 eves. or write to 16 Oakworth Croft, Moorside, Oldham, OL4 2LE.

BBC Master 128, BBC software, double switchable disc drives, colour monitor, EPROM blower, EPROM eraser, Sideways ROM box, Mouse and joystick. The software is disc (DFS and ADFS) or ROM based, all originals, too many to mention here. Detailed list of all sale items available, write to: 7 Lamerton Close, Bordon, Hants, GU35 0HY.

BBC FE level computer literacy Open Learning Pack "Inside Information" reviewed in BEEBUG July 1988 issue. BBC + MS-DOS floppy discs, two audio cassettes, User Guide, Student Guide and 224 page book. Boxed complete, new. Cost £41.35, will sell for £28 post paid. Tel. 01-907 5443.

2 Cumana 40T disc drives each with PSU £50 each. Tel. (0475) 23332.

Master 512 with DOS 2.1 £350. Brand new Watford dual 5.25/3.5 disc drive in steel plinth with own power supply £130. Phillips 7502 hi-res green monitor £30. Master ref manuals 1 and 2 £10. Viglen Master cartridge system and 5 cartridges £10. BEEBUG ICON master and Master ROMs £15 each. All items in excellent condition complete with original packaging and documentation. Will sell as priced or the lot for £460. Tel. 01-200 3210 day, (0992) 716361 after 8pm.

Sanyo 14" colour data display monitor for use with BBC computer £95 o.n.o. Turbo 65c102 Co-processor as new £75 o.n.o. Tel. (0234) 267067.

Master series reference manuals part 1 and 2 £10 each. New Advanced User Guide £13 p&p £1 each. All new and unused. Tel. (0424) 813794.

Computer Concepts InterBase ROM, new, unwanted gift £35. Tel. (0303) 42540. eves.

NEW SOFTWARE

BASIC STATISTICAL LIBRARY

A Collection of Procedures for Acorn Microcomputers
D Cooke and A H Craven

The 60 procedures (and functions) in *Basic Statistical Library* are intended for use on BBC B, Master series and Archimedes microcomputers. It has two major uses: to simplify construction of statistical programs and to act as a statistical super-calculator, making use of the power of Acorn computers in immediate command mode.

Publication March 1989 £39.50 + VAT
3.5" disk (ADFS) ISBN 0 7131 3687 1
5.25" disk (DFS) ISBN 0 7131 3625 1

MICROTAB – AN ALL-PURPOSE STATISTICAL PACKAGE

P G Higginbotham

"... *Microtab* is a minor miracle of programming, packing more into the BBC than I would have thought possible. It has over 100 commands ... Microtab offers extensive data editing and manipulation, histograms, and scatter plots, tabulation, one-way and two-way analysis of variance ... a comprehensive and easy-to-use teaching package." *The Times Higher Education Supplement*

1985 Floppy disk* and 96 pages manual £39.50 + VAT

***Disks are availabe in two versions:**
80 track (package ISBN 0 7131 3608 1)
dual-format 40/80 (package ISBN 0 7131 3549 2)

Additional manuals are available from the publishers.
£4.95 net 96 pages ISBN 0 7131 3560 3

Additional information is available from the Promotion Department at the address below.

Edward Arnold
A division of Hodder & Stoughton
41 Bedford Square London WC1B 3DQ Telephone: 01-637 7161

BBC B with OPUS DDOS, AMX Mouse £225. Also ROMs (including; Spellcheck, Fileplus Database, ROMit, Pascal, from £10). Games and Discs (Elite, Chelo, etc from £6). Tel. (0525) 717131 for full list.

Edword Superpack £25, Edspell (for Edword 80T) £10, Multi-Font NLQ £12, Wordwise £5, Windowmatic/ Technomatic £5,Dotprint Plus/ Technomatic £5, Mini Office II 80T + Dabs Mini Driver/Dabhand Guide £20, BEEBUG Spellcheck III£15, AMX Stop Press £20, AMX Super Art £20, Watford Quest Mouse £12. Tel. 01-399 2865.

Microlink Multi Speed Modem with cables and Microlink communication software. Excellent condition. £120. Tel. (0753) 652332.

System Delta with card index (BBC version) complete SD Personal Accountant package. Purchased Nov '88. £170 o.n.o. Tel. (0732) 863522.

Master Turbo, BEEBUG Master ROM, Overview cartridge, ISO Pascal, all manuals, BEEBUG and other magazines £400. Tel. (061) 483 2746.

BBC B 1.2OS with Watford DDFS and Vine Micros Replay £210. Watford 40/80T single disc drive £60. Tel. 061 962 9068.

Watford 32k ROM RAM board plus miscellaneous items including textbooks, joysticks, speech synthesiser, software £30 the lot. Tel. (044282) 4600.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES		BEEBUG & RISC USER
£ 7.50	6 months (5 issues) UK only	
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£23.00
£20.00	Rest of Europe & Eire	£33.00
£25.00	Middle East	£40.00
£27.00	Americas & Africa	£44.00
£29.00	Elsewhere	£48.00

BACK ISSUE PRICES (per issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

POST AND PACKING
Please add the cost of p&p as shown opposite.

BEEBUG
Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Advertising: Sarah Shrive
Production Assistant: Sheila Stoneman
Membership secretary: Mandy Mileham
Editorial Consultant : Lee Calcraft
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1989
ISSN - 0263 - 7561
Printed by Head Office Design (0782) 717161

Magazine Disc/Cassette

MARCH 1989 DISC/CASSETTE CONTENTS

3D LANDSCAPES - two programs, one for the Master and Compact and one for the model B, to generate 3 dimensional landscapes.

PAGE COMPOSITION FOR THE BBC MICRO (Part 1) - a pair of programs to assist in the design and layout of an A4 text page.

DUAL COLUMN LISTINGS - a utility to print (or display) automatically Basic program listings in two columns on the page.

CENTRES OF GRAVITY - both the CREATE and DISPLAY programs together with a sample data file (PERSON) to experiment with.

ADFS WIPE UTILITY - use this utility to add a *WIPE command to your ADFS filing system.

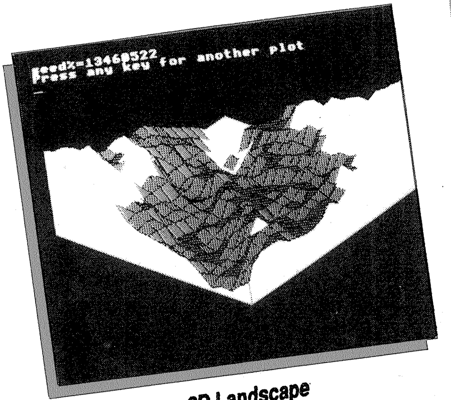
MULTI-PRECISION DECIMAL ARITHMETIC (Part 1) - a program and a demonstration of the first stages in implementing multi-precision arithmetic in assembler.

AN AUTO-DATING UTILITY - a short utility allowing the current date and time to be saved automatically in a program or a View file. A second program combines this with the previous Auto-Save utility.

FIRST COURSE - a short demonstration of timing.

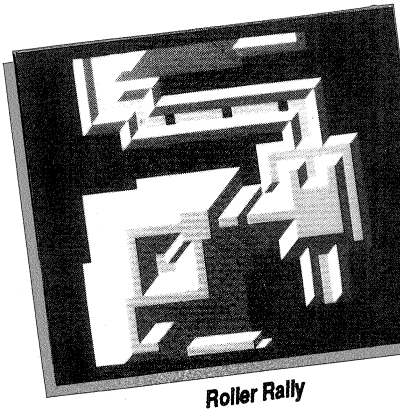
ROLLER RALLY - a highly challenging game which will test your keyboard skill and concentration to the full. A second version of this game, called Roller Coaster, is also included on the disc/tape.

MAGSCAN - bibliography for this issue (Vol.7 No.9).



```
Seed%=13460522
Press any key for another plot
```

3D Landscape



Roller Rally

SUBSCRIPTION RATES	UK ONLY			OVERSEAS		
	5" Disc	3.5" Disc	Cassette	5" Disc	3.5" Disc	Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

BEEBUG Discs - The Ultimate in Quality & Reliability

80 Track Double Sided Quad Density

	Members Price	Order Code
10	£9.90	0660
25	£24.90	0664
50	£39.90	0668

Free disc boxes

25/50 discs with free lockable storage box

Prices shown are members prices and include VAT.

BEEBUG discs are manufactured to the highest specifications and are fully guaranteed.

POSTAGE

Boxes of ten discs £2 Post code C

Boxes of 25 or more £4 Post code E

40 Track Single Sided Double Density

	Members Price	Order Code
10	£8.90	0657
25	£21.85	0661
50	£35.60	0665

3.5" Double Sided Double Density

	Members Price	Order Code
10	£15.00	0675
25	£56.00	0676

Our Guarantee

We confidently offer a lifetime data guarantee and will replace any disc with which you encounter problems. We have found that the standard of quality control at the factory makes this necessity very rare.

Please send me ______ (qty) ______ (stock code) at £______ (unit price)

UK post 10 £2, 25/40/50 £4. See retail catalogue for Overseas postage rates

I enclose a cheque for £______ /Please debit my Access/Visa card £______

| | | | | | | | | | | | | | | | | Expiry date: | | |

Name ____________________ **Memb No.** ____________________

Address ____________________

____

BEEBUG
Dolphin Place,
Holywell Hill,
St. Albans,
Herts.
AL1 1EX

☎ (0727) 40303