AMSTRAD AMSTRAD AMSTRAD AMSTRAD

Computing with the AMSTRAD

AMTIX!

$4.50

The independent AUSTRALIAN magazine for Amstrad users
Registered by Australia Post – Publication No. TBP 1053

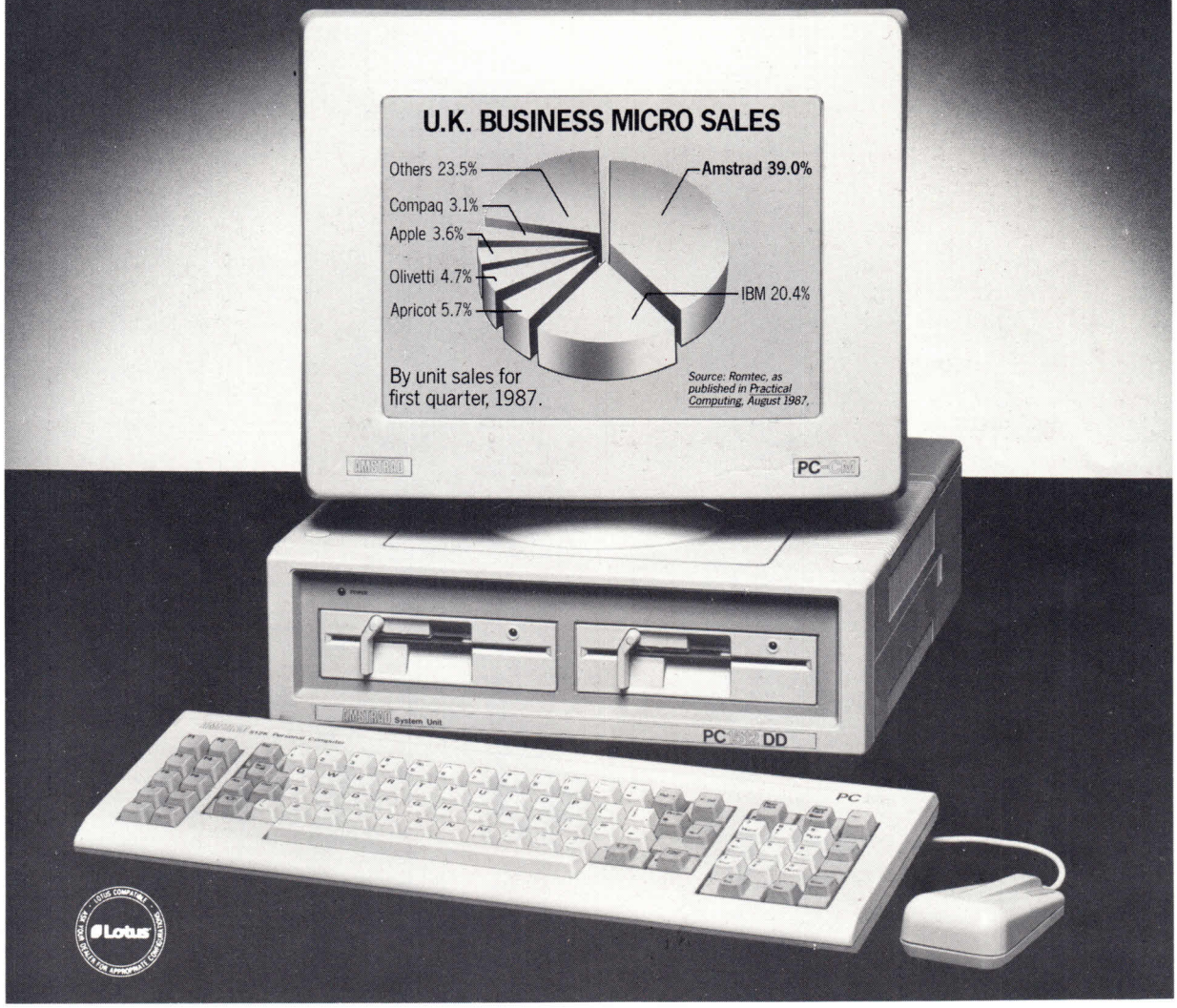
DECEMBER, 1987
A Database Publication

- SANTA'S SLEIGH – GAME OF THE MONTH
- EASY DRAW – DRAW ON YOUR SCREEN
- BIG SAVINGS ON BACK ISSUES OF MAGAZINES
- GREAT SPECIALS ON SOFTWARE

$ SAVINGS ON MAGAZINES $

AMSTRAD – AMSTRAD – AMSTRAD

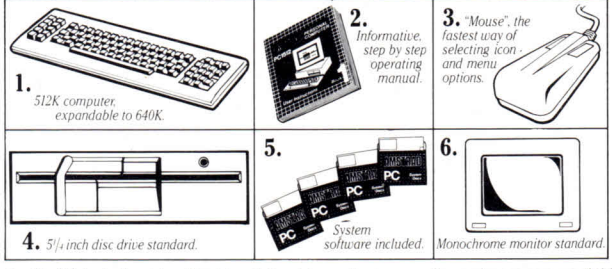
Why England's top selling PC is a better choice for your business.



A few years ago, Amstrad developed an idea which has made their Personal Computer the number one seller in the United Kingdom and Europe.

The idea, put simply, was to create the best value PC in the world. That's why every Amstrad PC comes complete with everything you need to make it useful.

A total PC* package from around $1499.



Double 5¼ inch disc drive, 20mb hard disc drive and colour monitor packages also available.

Value for money. The essence of a great idea.

Operating software and a monitor, expensive extra's on many PC's, are standard features of the Amstrad package. As are the mouse, keyboard, disc drive and an informative manual.

Added to this, Amstrad PC's are IBM* compatible. Thus, the capabilities of your system are virtually endless because you have access to thousands of excellent programs. You have the flexibility to perform the most advanced to the simplest of computing tasks.

All this for about half what you'd expect to pay. And fully backed by one of Australia's foremost electronics companies.

Distributed and guaranteed by Mitsubishi Electric AWA.

Amstrad is distributed through an Australia-wide dealer network and guaranteed for twelve months by Mitsubishi Electric AWA. Contact your Amstrad dealer today and see why Amstrad is a better choice for your business.

For the name of your nearest dealer, telephone: **Sydney** (02) 638 8444; **Melbourne** (03) 357 1688; **Brisbane** (07) 277 0100; **Perth** (09) 277 7788; **Adelaide** (08) 340 1033; **Hobart** (002) 72 4366.

**IBM is the trademark of International Business Machines Corp.*

AMSTRAD

Built on a great idea.

TPB 1618

Contents

Published monthly by Planet Publishing Pty Ltd
under licence from Database Publications Ltd.
Mail: P.O. Box 11, Blackmans Bay, Tasmania 7052.
Telex: AA58134, Attn. HT163 General enquiries, phone (002) 29 4377

PAINTING BY NUMBERS

IAN SHARPE expands the idea of mathematical trees to produce some attractive landscapes

Computing with the Amstrad readers are forever asking for more articles on graphics, so to help fill your insatiable appetites, here's more.

Trees caught my attention in *Microcomputer Graphics* by Mike Batty (Chapman Hall 1987 ISBN 0 412 28540 1). The author introduces simple graphics routines and develops them into interesting picture generating programs.

Poor old Mike uses the BBC Micro for his examples (we all have our crosses to bear), but BBC Basic does have procedures which can be called recursively with local variables and Mike takes advantage of these features to generate his trees.

Of course we don't have procedures on the Amstrad and using Robin Nixon's bolt–on routine from the May 1987 issue would have been admitting defeat. So I set about writing my own tree routine to get round the problems and came up with Program I. It's not an adaptation of the BBC Micro version but a different way of producing the same results.

The idea is that the tree is composed of branches, and at the end of each two new branches grow. At the tip of the new branches are two more and so on. You can see what I mean in Figure I.

Before we see how it works try it out for yourself. You will be prompted for three parameters – angle, depth and size. Angle is the angle of the V between two branches, depth is the number of times the tree is to subdivide and size is the length of the stem between its start and where it first splits into two. To begin with, try values of 45,8 and 110 respectively. Also have a look at 120,10,150,180,10,200 and 270,10,200.

What happens is that every time

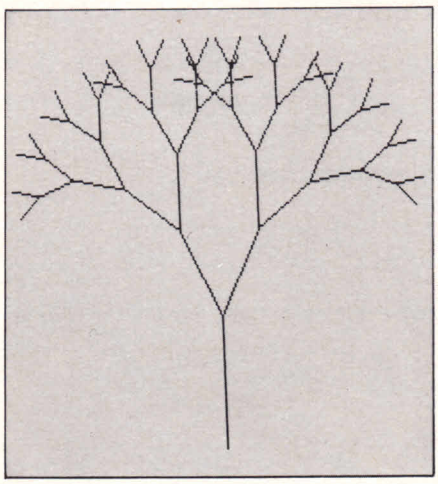


Figure I: A simple tree

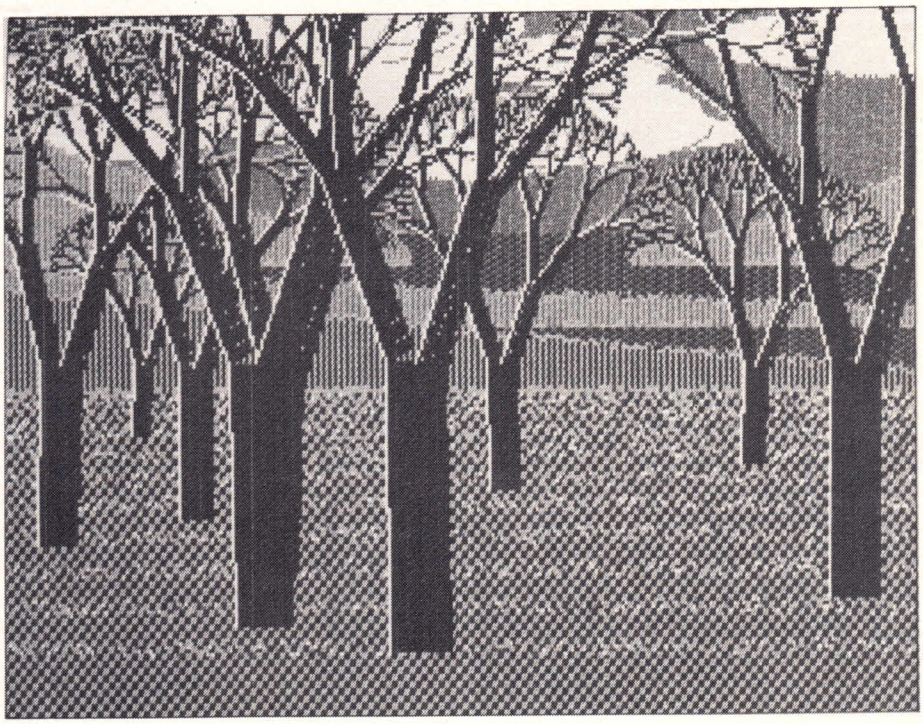


Figure II: Typical landscape created by Program II

a V is drawn the program calculates the coordinates of the tips. It takes the right–hand fork, draws a line to the tip and in the array *stack* it stores the details – size, coordinates, angle – of the left–hand tip which it can't deal with yet. It repeats the process, always taking the right fork and storing the left until it has gone to the required level held in *depth*.

If you think about it, the maximum number of left shoots stored in stack can't be more than *depth*. When the required number of shoots have been drawn by taking right turns, the routine backtracks in *stack* to the details of the last left turn it missed. The program uses a pointer –sp – to keep track of which part of *stack* it is taking its parameters from.

It follows this new route, again always forking right and remembering places it needs to come back to. In this way nothing is left out and eventually the entire tree is plotted. At this point there are no more parameters stored in *stack*. The program detects this and comes to an end.

To obtain the trunk, *stack* is initialised with just enough information for one leg of the first V.

Program II – Landscape – uses a modified tree routine to introduce thickness, colour and some random elements to the branches which gives an impression of a real tree with spring foliage.

Using more ideas borrowed from Mike Batty other parts of Program II draw clouds, water, hilly background and a foreground. Again these are controlled by overall rules with random factors, so a different but convincing arrangement emerges every time the program is run.

One such scene is shown in Figure II and I think you'll agree that it isn't bad for something created by a few dozen lines of Basic.

I don't make any claims for its artistic merit but it's far better than I could achieve by hand. Landscape isn't perfect, but it's fairly simple and like many of the programs we publish, it's the starting point for your own ideas.

This program illustrates one of the fundamental points in Microcomputer Graphics. Mike Batty is at pains to highlight the distinction between art on a computer and computer art. The first is where the artist uses a drawing package to create pictures on the screen.

He isn't doing anything different to his colleague working with paints, just using a different medium, though the coarseness of a home computer display makes the level more akin to embroidery or tapestry.

In the second case, computer art, the computer generates the picture from a model programmed by the artist. This is a new concept brought about by the availability of machines which are programmable and have good display capabilities. It opens up a new world of creative graphics to conventional artists and those who have artistic inclinations but may lack the manual skills.

I'll leave you to ponder two questions that puzzle a lot of people – is computer art really art and if so, does the art lie in the program or in the display on the screen?

If you want to see some impressive graphics created on main–frames with very high resolution displays, ask at your library for *Creative Computer Graphics* by Jankel and Morton (Cambridge University Press 1984 ISBN 0 521 26251 8). It'll remove any doubts on the first point and give you food for thought on the second.

```
10 REM        Program I - Trees
20 REM        by Ian C. Sharpe
30 REM (c) Computing With The Amstrad
40 REM-------------CPC---------------
50 DEFINT a-z:DEG:MODE 1
60 INPUT "Angle,Depth,Size? ",angle,depth,size
70 DIM stack(depth,3):stack(sp,0)=
0:stack(sp,1)=0:stack(sp,2)=angle/
2:stack(sp,3)=depth+1
80 CLS:ORIGIN 320,10:sp=0:i=1
90 REM-------------------------------
100 f=stack(sp,3)-1:IF f=0 THEN sp
=sp-1:IF sp=-1 THEN END ELSE 100
110 x=stack(sp,0):y=stack(sp,1):a=
stack(sp,2)
120 s=size*0.75^(depth-f-1)
130 a(0)=a-(angle/2):a(1)=a+(angle/2)
140 tx=x+s*SIN(a(0)):ty=y+s*COS(a(0))
150 MOVE x,y:DRAW tx,ty
160 GOSUB 190:GOSUB 190
170 sp=sp+1:GOTO 100
180 REM--------------------------
-----
190 i=i XOR 1
200 stack(sp+i,0)=tx:stack(sp+i,1)
=ty
210 stack(sp+i,2)=a(i):stack(sp+i,
3)=f
220 RETURN
```

Program I: The basic tree routine

```
10 REM       Program II - Landscape
20 REM            by Ian C. Sharpe
30 REM (c) Computing With The Amst
rad
40 REM-------------------------
----
50 DEFINT a-z:DEFREAL r,w
60 RANDOMIZE TIME:DEG:DEF FNrn(r)=
CINT(RND*r)
70 MODE 1:BORDER 1:INK 0,0:INK 1,1
1:INK 2,18:INK 3,26
80 REM --- Sky & big fluffy clouds
 ---
90 PAPER 1:CLS:PAPER 0
100 FOR clouds=1 TO 5+FNrn(2)
110 cloudx=30+FNrn(580):cloudy=330
+FNrn(10*clouds)
120 FOR fluffybit=1 TO 4+FNrn(2)
130 ex=cloudx-50+FNrn(100):ey=clou
dy-30+FNrn(60)
140 er1=60+FNrn(8*clouds):ratio=0.
1+RND*0.2
150 GOSUB 380:NEXT:NEXT
160 REM ---- Sea, ground & hills -
---
170 LOCATE 1,9:PRINT STRING$(200,1
27);
180 PEN 2:PRINT STRING$(240,206);S
TRING$(240,206);:PEN 1
190 FOR dy=0 TO 12:y=182-dy*dy
200 FOR x=0 TO 638 STEP 2:PLOT x,y
+FNrn(8),1+FNrn(2)
210 NEXT x,dy
220 FOR hills=1 TO 3+FNrn(1)
230 hy=290-hills*35:ht=(6-hills)*1
1:wlhl=(0.15+RND)/3:lift=ht+FNrn(h
t/10):clip=FNrn(ht/6):offst=FNrn(2
00)
240 col(0)=2:col(1)=FNrn(1)
250 GOSUB 450:NEXT
260 REM -------Grow some trees ---
----
270 FOR size=40 TO 160 STEP 15:RAN
DOMIZE TIME
280 treey=200-size:treex=20+FNrn(6
00):treewid=CINT(size/4)
290 IF TEST(treex,treey+30)<>2 THE
N 280
300 angle=FNrn(6)-44:depth=8+FNrn(
(size-40)/40)
310 DIM stack(depth,3)
320 stack(0,0)=0:stack(0,1)=0:stac
k(0,2)=angle/2:stack(0,3)=depth+1
330 ORIGIN treex,treey:sp=0:i=1
340 GOSUB 500:ORIGIN 0,0:ERASE sta
ck:NEXT
350 REM = Press Space when finishe
d =
360 PRINT CHR$(7);:WHILE INKEY(47)
<>0:WEND:RUN
370 REM ==== Elipse for clouds ===
=
380 ORIGIN ex,ey:ersq=er1*er1
390 FOR x=0 TO er1 STEP 2
400 y=2*ratio*SQR(ersq-x*x)
410 MOVE -x,-y/2:DRAWR 0,y,3:PLOTR
 0,0,1:IF RND<0.25 THEN PLOTR 0,-y
420 MOVE x,-y/2:DRAWR 0,y,3:PLOTR
0,-y,1
430 NEXT:ORIGIN 0,0
440 RETURN
450 REM -------Sine wave Hill-----
--
460 FOR x=0 TO 638 STEP 2
470 hh=MAX(lift+ht*SIN(offst+(x*wl
hl)),clip)
480 IF hh>4 THEN MOVE x,hy:DRAWR 0
,FNrn(4):DRAWR 0,hh,col(z):z=z XOR
 1:PLOTR 0,0,2
490 NEXT:RETURN
500 REM-----------Tree-----------
510 f=stack(sp,3)-1:IF f=0 THEN sp
=sp-1:IF sp=-1 THEN RETURN ELSE 51
0
520 x=stack(sp,0):y=stack(sp,1):a=
stack(sp,2)
530 df=depth-f:s=size*(0.7^df):bwi
d=2+treewid/(2^df)
540 a(0)=a-(angle/2)+FNrn(4)-2:a(1
)=a+(angle/2)+FNrn(4)-2
550 tx=x-s*SIN(a(0)):ty=y+s*COS(a(
0))
560 xd=tx-x:yd=ty-y
570 ix=bwid*COS(a(0)):IF ix=0 THEN
 ix=1
580 xs=SGN(ix):iy=bwid*SIN(a(0))
590 FOR xx=x+ix TO x STEP -xs:yy=y
+iy*(xx-x)/ix:MOVE xx,yy:DRAWR xd,
yd,0:NEXT:DRAW x,y,3+(f<FNrn(3)+de
pth*0.35)
600 GOSUB 630:GOSUB 630
610 sp=sp+1:GOTO 510
620 REM-------------------------
-----
630 i=i XOR 1
640 stack(sp+i,0)=tx+i*ix/2:stack(
sp+i,1)=ty+i+iy/2
650 stack(sp+i,2)=a(i):stack(sp+i,
3)=f
660 RETURN
```

Program II: Landscapes unlimited

Still carrying the flag

Part IX of MIKE BIBBY's introduction to machine code

Last month we spent a lot of time looking at the Carry flag – particularly how it could be used together with CP to create loops, filter inputs and so on.

However, the Carry flag isn't the only flag on the Z80: There are six altogether. The one we're interested in at the moment, though, is the Zero flag.

Actually, the Zero flag's name virtually tells you what it does – the Zero flag is set when the "answer" to some machine code operation comes to zero.

For example:

```
LD A,&FF
ADD A,&Ø1
RET
```

Program I

will not only leave 0 in the A register, it will also set the Zero flag to tell us the result of the operation was zero.

Similarly:

```
LD A,&DF
SUB &DF
RET
```

Program II

will both leave zero in A and set the Zero flag.

So, if an ADD or SUB leaves zero in the A register, the Zero flag is set. If the result is non–zero, the Zero flag is cleared.

If you think about it, it's rather topsy–turvy: When the result is zero you set the Zero flag – to one. And when the result is not zero, you clear the Zero flag – to zero. One when it's zero, zero when it's not!

Don't worry about it – the actual values of flags don't really concern us. All we need to know is what conditions set and clear the flag. In this case the Zero flag is set when the result of an operation is zero, and cleared when not.

Not every instruction affects the Zero flag, but you'll soon get used to the ones that do – they're fairly obvious.

You'll probably remember that we had a pair of jump instructions based around the Carry flag – JP C and JP NC.

Well, we have a corresponding pair of jump instructions involving the Zero flag.

JP Z (opcode &CA)

performs the jump if the Zero flag has been set, and

JP NZ (opcode &C2)

takes the jump if the Zero flag is not set – that is, it's clear.

Take a look at Program III. It shouldn't present you with any problems, we've met it twice before: It was our first loop.

address	hex code	mnemonics
3ØØØ	3E 2Ø	LD A, &2Ø
3ØØ2	CD 5A BB	CALLCharOut
3ØØ5	C6 Ø1	ADD A,1
3ØØ7	D2 Ø2 3Ø	JP NC,&3ØØ2
3ØØA	C9	RET

Program III

You'll remember that we keep on incrementing the A register and jumping back to print out the corresponding Ascii code until we go round the clock – and in so doing set the Carry flag.

However, at the same time, the Zero flag will be set since the last ADD A,1 adds 1 to the 255 (&FF) in the A register, giving us zero.

This means we could rewrite Program III with a JP NZ, &3002 in place of the JP NC, &3002 – and this is what we do in Program IV

address	hex code	mnemonics
3ØØØ	3E 2Ø	LD A,&2Ø
3ØØ2	CD 5A BB	CALL CharOut
3ØØ5	C6 Ø1	ADD A,1
3ØØ7	C2 Ø2 3Ø	JP NZ,&3ØØ2
3ØØA	C9	RET

Program IV

Here we keep on performing the loop until the Zero flag is set by the ADD A,1 when A contains &FF. We then drop through the loop and RET.

Don't get the idea that the Carry and Zero flags are linked in any way. In the above example it's just coincidence that both are set at the same time – it isn't always so.

To prove it, take a look at Program V. We're using INC A (&3C) to increase A by 1 instead of the ADD A,1. Otherwise it's identical to Program IV, but one byte shorter.

address	hex code	mnemonics
3ØØØ	3E 2Ø	LD A,&2Ø
3ØØ2	CD 5A BB	CALL CharOut
3ØØ5	3C	INC A
3ØØ6	C2 Ø2 3Ø	JP NZ, &3ØØ2
3ØØ9	C9	RET

Program V

As you'll see, it works fine.

However, if, as in Program VI, we replace the JP NZ with JP NC – which we might do if we thought the Zero and Carry flags were linked – disaster occurs.

The point is that INC A sets or clears the Zero flag depending on the result of increasing A by one. However, INC A doesn't affect the Carry flag at all.

This means the Carry flag is not set when A is increased from 255 to 0. The loop therefore keeps going, effectively printing out CHR$(1), CHR$(2) and so on.

However the Ascii codes under 32 (&20) are control codes, which as you'll see from the User Guide, have some pretty powerful effects.

address	hex code	mnemonics
3ØØØ	3E 2Ø	LD A,&2Ø
3ØØ2	CD 5A BB	CALL CharOut
3ØØ5	3C	INC A
3ØØ6	D2 Ø2 3Ø	JP NC, &3ØØ2
3ØØ9	C9	RET

Program VI

In practice, your screen will lock up: your micro has crashed. Don't worry, you haven't hurt it – it's just that you'll have to reset the machine to regain control. Unfortunately, in doing so, you'll lose whatever program is in memory – in my case Hexer, my hexadecimal loader.

My advice is that you don't run Program VI until the end of the session, when you're ready to switch off anyway ... It certainly demonstrates the fact that the Carry and Zero flags are independent – in a way you're not likely to forget.

Actually, despite the obvious error in Program VI, there's a subtler one. We've assumed that the Carry flag is clear when we enter the program.

Think about it. If, when we run the program – either from Hexer or directly by CALL &3000 – the Carry flag were already set, it would remain so throughout the first three instructions.

As we've said, LD instructions don't affect the Carry flag, so LD A,&20 will leave Carry set, as will the call to CharOut. And, as we've already mentioned, INC A doesn't affect it.

This means that when we reach the JP NC, we don't take the jump, since Carry has been set from the beginning. We would then exit our routine via the final RET, having just printed out a space (Ascii &20).

In practice this doesn't happen. When we enter the routine, Carry is clear, and remains so since none of the instructions affects it. Hence the JP NC is always taken and we crash.

Still, in machine code, it's wrong to assume what the state of a particular flag will be. Doing so often causes elusive bugs in programs.

So in Program VI, we shouldn't just hope that Carry will be clear, we should force the issue. It would be nice if we could do this with a single Z80 instruction. Unfortunately, no such instruction exists.

However we can clear Carry with:

ADD A,Ø (C6 ØØ)

After all, you can't go round the clock by adding zero to anything, can you? And ADD does affect the Carry flag, so in this case Carry must be cleared. The added bonus (no pun intended) is that the contents of the A register remain unchanged – think about it.

Thus the effect of ADD A,O is to clear the Carry flag for us, without affecting the contents of the A register.

Right, we know how to clear the Carry flag when we want to – so how do we set it?

Well, this time there is an instruction that fits the bill – SCF (opcode &37). SCF stands for Set Carry Flag.

Program VII shows a trivial example of the use of these ideas.

address	hex code	mnemonics
3ØØØ	3E Ø7	LD A,7
3ØØ2	C6 ØØ	ADD A,Ø
3ØØ4	DA ØØ 3Ø J	P C, &3ØØØ
3ØØ7	37	SCF
3ØØ8	D2 ØØ 3Ø	JP NC, &3ØØØ
3ØØB	CD 5A BB	CALL CharOut
3ØØE	C9	RET

Program VII

As you can see, there are two conditional jumps which take you back to the beginning of the program. I've used ADD A,0 before the first and SCF before the second to rig things so that the jumps are not taken.

We go straight through to CALL CharOut, which gives us a bleep. If we get any of our conditional jumps the wrong way round, or if we set when we should clear, we'll get stuck in an infinite loop.

Program VIII shows another rather contrived example. If you follow through its rather tortuous path you'll see that, because of the way we've manipulated the Carry flag, nothing gets printed out.

address	hex code	mnemonics
3ØØØ	3E 2A	LD A,&2A
3ØØ2	C6 ØØ	ADD a,Ø
3ØØ4	D2 ØA 3Ø	JP NC, &3ØØA
3ØØ7	CD 5A BB	CALL CharOut
3ØØA	37	SCF
3ØØB	DA 11 3Ø	JP C,&3Ø11
3ØØE	CD 5A BB	CALL CharOut
3Ø11	C9	RET

Program VIII

Can you alter things so that you do indeed get two asterisks (Ascii code &2A) appearing on the screen? Hint: Have a good look at the conditions attached to those JP instructions.

Right, back to the Zero flag. Last month we saw that the CP n instruction, while not altering the A register, set and cleared the Carry flag as if a SUB n had been performed:

Carry was set if A < n
Carry was clear if A >=n

You won't be surprised to learn that CP n also affects the Zero flag as well:

If the byte in the A register is identical to n, the zero flag is set.

If the two bytes differ, the zero flag is cleared.

This is only what you'd expect. If the two bytes are the same, when the CP n does its simulated A – n, the answer must be zero, so the Zero flag's set.

If, on the other hand, the bytes vary, the answer is non–zero and the flag is cleared.

We can use this to "demand" that a certain character be input from the keyboard, as in Program IX, where we wait for an asterisk:

address	hex code	mnemonics
3ØØØ	CD 18 BB	CALL Charin
3ØØ3	FE 2A	CP &2A
3ØØ5	C2 ØØ 3Ø	JP NZ, &3ØØØ
3ØØ8	CD 5A BB	CALL CharOut
3ØØB	C9	RET

Program IX

The program works by comparing the input character with the Ascii for an asterisk. If it is an asterisk, the zero flag is set and, ignoring the jump, we print it out and return.

If it isn't an asterisk, the codes will differ and the Zero flag will be cleared. We then jump back to the beginning, to get another character from the keyboard.

So far, when using CP, we've compared the A register with a specific number. We can, however, compare the A register with any other of our single registers, as Table I shows. The notation for this form of instruction is CP r, where r is an eight bit register.

CP B, for example, compares the byte in the A register with that in the B by doing a dummy A–B and setting the flags accordingly – the contents of both registers remain unchanged. Note, the register specified is subtracted from the A register.

Again, if A is equal to or greater than B, Carry is clear. If A is less than B, Carry is set. Also, if the bytes in A and B are equal, the Zero flag is set, otherwise it's cleared.

We put this ability to compare our eight bit registers to use in Program X. Here our task is to print out a fixed number of asterisks – eight in this case.

```
address hex code    mnemonics
3ØØØ    Ø6 ØØ      LD B,Ø
3ØØ2    3E 2A       LD A,&2A
3ØØ4    CD 5A BB    CALL CharOut
3ØØ7    Ø4          INC B
3ØØ8    3E Ø8       LD A,&ØB
3ØØA    B8          CP B
3ØØB    C2 Ø2 3Ø    JP NZ,&3ØØ2
3ØØE    C9          RET
```

Program X

We use the B register as a counter. Initially we set it to zero with LD B,0 and then print out an asterisk by loading A with &2A and calling CharOut. We next increase B, thus keeping track of the number of asterisks printed. We then load A with 8, the number of asterisks, then compare the B register with A.

If the two registers aren't equal – that is, if the Zero flag is not set – we haven't printed eight asterisks so we jump back to &3002, reloading A with &2A, calling CharOut and so on.

If, on the other hand, we've reached our limit, the Zero flag is set and the jump isn't taken so we simply return.

```
address hex code    mnemonics
3ØØØ    Ø6 Ø8       LD B,8
3ØØ2    3E 2A       LD A,&2A
3ØØ4    CD 5A BB    CALL CharOut
3ØØ7    Ø5          DEC B
3ØØ8    C2 Ø4 3Ø    JP NX,&3ØØ4
3ØØB    C9          RET
```

Program XI

Actually, the program is rather tortuous. There are more efficient ways to print out a number of asterisks. I just wanted to introduce the CP r instruction. Program XI shows an alternative way of doing it. The trick is to count down from the number you want by using DEC B.

When we reach zero, the Zero flag is set, otherwise we jump back and print another asterisk. In effect, B acts as a primitive loop variable or counter.

You can use this idea to create nested loops – using the B register for the inner loop and another for the outer loop.

Program XII uses this idea to print out a triangle of asterisks:

```
address  hex code    mnemonics
3ØØØ     ØE Ø8       LD C,&Ø8
3ØØ2     41          LD B,C
3ØØ3     3E 2A       LD A,&2A
3ØØ5     CD 5A BB    CALL CharOut
3ØØ8     Ø5          DEC B
3ØØ9     C2 Ø5 3Ø    JP NZ,&3ØØ5
3ØØC     3E ØA       LD A,&ØA
3ØØE     CD 5A BB    CALL CharOut
3Ø11     3E ØD       LD A,&ØD
3Ø13     CD BB 5A    CALL CharOut
3Ø16     ØD          DEC C
3Ø17     C2 Ø2 3Ø    JP NZ,&3ØØ2
3Ø1A     C9          RET
```

Program XII

We're going to have eight lines, eight asterisks in the first, seven in the second and so on. So each time round the outer loop we'll print a line of asterisks – the inner loop printing out the required number each time.

We use C to count the number of lines – hence the initial LD C,&08 – and B the number of asterisks. When you think about it, since we're using it as a counter, C goes down by one each time round the outer loop which is also what we want to happen to B. Our second instruction, the beginning of the outer loop, is therefore LD B,C.

If you look towards the end of the program, you'll see the end of the first loop:

```
3Ø16    ØD          DEC C
3Ø17    C2 Ø2 3Ø    JP NZ,&3ØØ2
```

To return to the start of the outer loop, after LD B,C we load the A register with the Ascii for asterisk (LD A,&2A). We then arrive at the inner loop.

```
3ØØ5    CD 5A BB    CALL CharOut
3ØØ8    Ø5          DEC B
3ØØ9    C2 Ø5 3Ø    JP NZ,&3ØØ5
```

As you'll soon see, this prints out "B asterisks". Now the value of B goes down by one each time through the outer loop since we get it via the C register (LD B,C) and C is decreased by the DEC C at the end of the outer loop each time. This means that each line has one less asterisk than the previous line – giving us our triangle effect. When C gets to zero – after we've printed eight lines – the program drops through the JP NZ,&3002 of the outer loop and RETs. Incidentally the lines between the JP NZs of the inner and outer loops.

continues page 24

Max Review

By SHANE KELLY

My computer is a CPC 464 with a ram expansion. I, like most of my computer owning friends, am always after a larger more powerful computer. My current passion is for WIMP'S (no smart remarks, please!) and the APPLE MACINTOSH is the standard of excellence in this field. I drool when I see the power available at the click of a mouse's button.

So, when a product dropped across my desk with the comment from the supplier that it was WIMP'S for Amstrads I naturally had to look deeper into it.

Basically speaking, AMX'S new product MAX is a friendly front end for AMSTRADS. It has pull–down menus and icons and windows and you can use your mouse should you desire (or keyboard or joystick!) to do all those little chores that had to be done by typing ***ERA,@A$ etc. OK, I thought (which I don't do very often) these WIMP'S are supposed to be intuitive so I shouldn't have to read the manual to get started. What happened was this:–

– The screen opened to show the desktop. Intuitively, I pressed the cursor keys. Hey presto, the pointer moved. When I got the pointer over the disc icon I then pressed the return button. Nothing happened. Puzzled but undaunted (and having some experience with AMX PAGEMAKER) I pressed function key 4 (the 'execute' button in PAGEMAKER) and guess what? Nothing happened. OK, time to hit the books. Turn to page one of your MAX manual. The first thing to read is the warning about destroying discs which can happen to those unfamiliar with the system.

• After reading that I thought (again?) that it was time to sit down and really study the manual as you should do before you use any piece of new software.

The manual is clearly and logically laid out and progresses from first principles through to an explanation of each item on the desk top, which even includes notes on the shortcomings of some of the utilities supplied. For instance, under the copying of files there are several notes about the length of files that can be copied and what happens if there is an error. Nice touch this as some software just dumps you back into the operating system and your files are in limbo.

The MAX package contains the following:

a) the MAX desktop or control panel. It is here that the MAX environment is altered to suit your individual taste and working habits. A fair amount of control is given:— you may choose your colour combinations and your keys along with the speed of the pointer and these parameters may be saved so that they are available on start up.

b) Windows – while these are not as flexible as the MACS they are resizeable and moveable and you are able to have more than one on the screen at any one time.

c) Utilities – these are a formatter, sector editor, file copier and printer control (to stop the double line feed problem). Fairly comprehensive instructions on all these options are given.

MAX is a unique package made up of standard utilities presented in a manner that is novel (for CPC's) and fun to use. It is robust and thoughtful and I have found that it will not upset your delicate stomach by crashing at exactly the wrong time.

I really don't know where to put MAX in the structure of things. I use it when I am in BASIC and have to shuffle files around or recover a file accidentally deleted. I use it when I feel the need to pretend that my computer is more expensive than it is, and I use it when I need to impress people who constantly refer to home computers as toys and so damn difficult to use. (This still happens quite often these days, I find). This justifies MAX's inclusion in my library of files and justifies it's purchase.

If you're looking for a piece of software that is truly useful, good to look at and written well enough to do what it has to without fuss then the unique MAX from AMX systems is for you.

The multi-coloured aliens are landing

Part II of ROLAND WADDILOVE's series on how to produce a better class of graphics using machine code

Last month we looked at how the Mode O screen memory was organised using a few simple programs ... Now we're going to try some short machine code routines to print a multi-coloured character on the screen.

First I'll briefly recap what we learnt last time.

The screen is organised into rows of pixels and a single character occupies 32 bytes of memory, eight rows of four bytes. The bit pattern of each byte in the screen memory holds the information for two horizontal pixels.

Going down the screen the rows of pixels are in groups of eight. There are 25 groups on the screen – these are the lines we LOCATE and PRINT on. The address of each row or pixels in the group is &800 more than the previous row and each group starts &50 more than the one above. Figure I shows the top left corner of the screen.

```
1Ø REM PROGRAM I
2Ø POKE &9ØØØ,8:POKE &9ØØ1,4
3Ø FOR i=Ø TO 31
4Ø READ j: POKE &9ØØ2+i,)
5Ø NEXT
6Ø REM alien
7Ø REM Rows=8/Columns=4
8Ø  DATA  4,12,12,8,72,148,1Ø4,
    132,72
9Ø  DATA  156,1Ø8,132,28,6Ø,
    6Ø,44,28
1ØØ DATA 32,16,44,4,48,48,8,
    8,Ø,Ø,4
11Ø DATA 4,Ø,Ø,8
```

Program I

To display a normal–size character on the screen all that is necessary is to work out the data required and store it in eight rows of four bytes somewhere in the screen RAM. To make it easy, first we'll print a normal–size character exactly on a line.

We need some character data, so run Program I to poke the data for an alien to &9001. This information will be used by the machine code routines.

Program II is an assembler listing of the routine to print the alien. You can either use an assembler to enter the mnemonics, or enter the hex codes one at a time using Program III.

To see Program II's print routine in action enter:

```
MODE Ø:?:?:?
CALL &8ØØØ
```

The alien is printed in the top left corner of the screen at &C000.

First BC is loaded with the number of columns and rows which is stored at &9000, DE with the address of the data, &9002 and HL the screen address where we want

```
RAW Assembler V.3
Pass ...        2 ORG &8ØØØ
                  PROGRAM II
8ØØØ:ED 4B ØØ 9Ø LD BC, (&9ØØØ)
8ØØ4:11 Ø2 9Ø LD DE,&9ØØ2
8ØØ7:21 ØØ CØ LD HL,&CØØØ
BØØA: .loop1
8ØØA:C5 PUSH BC
8ØØB:E5 PUSH HL
8ØØC: .loop2
8ØØC:1A LD A,(DE)
8ØØD:77 LD(HL),A
8ØØE:23 INC HL
8ØØF:13 INC DE
8Ø1Ø:1Ø FA DJNZ loop2
8Ø12:E1 POP HL
8Ø13:Ø1 ØØ Ø8  LD BC,&8ØØ
8Ø16:Ø9 ADD HL,BC
8Ø17:C1 POP BC
8Ø18:ØD  DEC C
8Ø19:2Ø EF JR NZ,loop1
8Ø1B:C9 RET
8Ø1C  END
```

Program II

```
1Ø REM PROGRAM III
2Ø a=&8ØØØ
3Ø PRINT HEX$(a);" : ";
4Ø INPUT h$
5Ø POKE a,VAL("&"+h$)
6Ø a=a+1:GOTO 3Ø
```

Program III

&C000	&C001	&C002	&C003	&C004	...
&C800	&C801	&C802	&C803	&C804	...
&D000	&D001	&D002	&D003	&D004	...
&D800	&D801	&D802	&D803	&D804	...
&E000	&E001	&E002	&E003	&E004	...
&E800	&E801	&E802	&E803	&E804	...
&F000	&F001	&F002	&F003	&F004	...
&F800	&F801	&F802	&F803	&F804	...
&C050	&C051	&C052	&C053	&C054	...
&C850	&C851	&C852	&C853	&C854	...
...	...	...	...	...	...

Figure 1 ... Memory map of top left of the Mode 0 screen RAM

it to be printed, &C000. B is the inner loop counter, the number of columns, and C is the outer loop the number of rows.

The loop counters and address of the row are first saved, then the inner loop runs along the row collecting the data pointed to by DE and storing it at the address pointed to by HL. HL and DE are then incremented to get the next data items and screen address.

At the end of the row the address of the start of the row is restored and &800 added to HL to get the address of the next row. The loop counters are restored and C, the number of rows, is decremented.

If you study the routine you'll see that the width of the character is irrelevant. It doesn't have to be four bytes wide – this just happens to be the width of our alien.

```
RAW Assembler V.3
Pass ... 2              DRG &8ØØØ

                        PROGRAM IV
8ØØØ:ED 4B ØØ 9Ø LD BC,(&9ØØØ)
8ØØ4:11 Ø2 9Ø LD DE,&9ØØ2
8ØØ7:21 ØØ EØ LD HL,&EØØØ
8ØØA:                   .loop1
8ØØA:C5                 PUSH BC
8ØØB:E5                 PUSH HL
8ØØC:                   .loop2
8ØØC:1A                 LD A,(DE)
8ØØD:77                 LD (HL),A
8ØØE:23                 INC HL
8ØØF:13                 INC DE
8Ø1Ø:1Ø FA              DJNZ loop2
8Ø12:E1                 POP HL
8Ø13:Ø1 ØØ Ø8           LD BC,&8ØØ
8Ø16:Ø9                 ADD HL,BC
8Ø17:3Ø Ø4              JR NC,ok
8Ø19:Ø1 5Ø CØ           LD BC,&CØ5Ø
8Ø1C:Ø9                 ADD HL,BC
8Ø1D:           .ok
8Ø1D:C1                 POP BC
8Ø1E:ØD                 DEC C
8Ø1F:2Ø E9              JR NZ,loop 1
8Ø21:C9                 RET
8Ø22:                   END
```

Program IV

```
RAW Assembler V.3
Pass ... 2              ORG &8ØØØ
                        PROGRAM V
8ØØØ:21 ØØ CØ           LD HL,&CØØØ
8ØØ3:Ø1 F4 Ø1           LD BC,5ØØ
8ØØ6:                   .next
8ØØ6:E5                 PUSH HL
8ØØ7:C5                 PUSH BC
8ØØ8:CD 17 8Ø           CALL print
8ØØB:C1                 POP BC
8ØØC:E1                 POP HL
8ØØD:23                 INC HL
8ØØE:23                 INC HL
8ØØF:23                 INC HL
8Ø10:23                 INC HL
8Ø11:ØB                 DEC BC
8Ø12:78                 LD A,B
8Ø13:B1                 OR C
8Ø14:2Ø FØ              JR NZ,next
8Ø16:C9                 RET

8Ø17:                   .print
8Ø17:ED 4B ØØ 9Ø LD BC,(&9ØØØ)
8Ø1B:11 Ø2 9Ø           LD DE,&9ØØ2
8Ø1E:                   .loop1
8Ø1E:C5                 PUSH BC
8Ø1F:E5                 PUSH HL
8Ø2Ø:                   .loop2
8Ø2Ø:1A                 LD A,(DE)
8Ø21:77                 LD (HL),A
8Ø22:23                 INC HL
8Ø23:13                 INC DE
8Ø24:1Ø FA              DJNZ loop2
8Ø26:E1                 POP HL
8Ø27:Ø1 ØØ Ø8           LD BC,&8ØØ
8Ø2A:Ø9                 ADD HL,BC
8Ø2B:3Ø Ø4              JR NC,ok
8Ø2D:Ø1 5Ø CØ           LD BC,&CØ5Ø
8Ø3Ø:Ø9                 ADD HL,BC
8Ø31:                   .ok
8Ø31:C1                 POP BC
8Ø32:ØD                 DEC C
8Ø33:2Ø E9              JR NX,loop1
8Ø35:C9                 RET
8Ø36:                   END
```

Program V

B is loaded with the width at the start and is decremented every time round the inner loop until it's zero. HL and DE are incremented to give the correct addresses.

What about the height? The alien is eight pixels deep and we're printing it at &C000. The outer loop adds &800 to the address in HL each time to get the start of the next row so the address of the last row is &F800.

Suppose that the character isn't exactly on the line. It might be printed at the fifth pixel down so that it's half on one line and half on the next.

We're going to have problems here because the character is split over two groups of eight rows. When we get down to the bottom row of a group adding &800 to the address will not give the address of the row which is the top of the next group of eight rows.

In Figure I &C050 is the address of the first row in the next group of eight rows, but &F800 + &800 will be 0. An overflow will occur because a register pair can only hold numbers up to &FFFF, if this is exceeded it wraps round the 0 again.

What we need to do is check to see if there has been an overflow, and if there has then add a correction factor – &C050. If there hasn't been an overflow we're OK.

Program IV is the same routine as before but an overflow check has been added. To test it we'll print our alien five pixels down, at &E000.

The screen is organised into rows of pixels and a single character occupies 32 bytes of memory, eight rows of four bytes. The bit pattern of each byte in the screen memory holds the information for two horizontal pixels.

The first four rows are OK, they start at addresses &E000, &E800, &F000 and &F800. Then there will be an overflow when &800 is added to HL for the fifth row. This sets the carry flag so &C050 is added to correct the result. Note that a check is made with each row and not just the fifth. This makes the routine general.

Again enter:

MODE Ø:?:?:?
CALL &8ØØØ

and you'll see the alien printed half on the first line and half on the second.

This short routine will now print any size multi–coloured character at any screen address. It doesn't matter whether it's exactly on a line or split over two or more, the code checks and corrects whenever necessary. Try it and see! Set HL to any value from &C000 on, assemble the routine again and call &8000.

It's difficult to get any idea of the speed advantage of the machine code routine over Basic when only one character is being printed. Program V completely fills the screen with aliens, and considering that each alien is made up of several different colours it's amazingly fast.

HL is used to store the address and BC is the loop counter. These are saved before printing the alien and restored afterwards. The print routine itself has been kept separate and has been labelled *print* for obvious reasons and is called as a subroutine. The HL register pair is used to pass the address to print the character.

• *I think that's enough to digest for this month. Next time we'll see how to get things moving.*

Now we're shaping up

IAN SHARPE continues his exploration into the fascinating world of turtle graphics

Last month I set you some homework that was similar to the group of squares we looked at – but there was a catch.

Although the internal angles are 60 degrees the turtle has to turn through the external angle of 120 degrees to draw them. If you realised that, you should have come up with something like:

```
cs
repeat 3 [rt 12Ø repeat 3 [ rt
    12Ø fd! 1ØØ ]]
```

Remember what I said about the exclamation mark – don't type it in because it's only there to show that the next line is a continuation of the one you're on.

It doesn't matter how long you made the sides of the triangles as long as you got the right idea. If you didn't manage to solve it by yourself read last month's article again and make sure you understand how it works before going any further.

Experimentation and practice are the keys to learning any language – computer or otherwise.

So far we have been entering instructions which the turtle follows as soon as we type them in. That's because we are working in direct mode which is like typing commands in Basic without line numbers. We need some way of writing Logo instructions which form a program that can be saved and run at a later date.

Logo doesn't need line numbers and wouldn't understand them if they were used. Programs in Logo are built up from procedures, which are lists of commands similar to the ones we have used to draw squares and triangles.

The difference is that they are grouped together under a name which can be used to recall and run them any number of times once the procedure has been defined.

Let's see how this works by using last month's square as an example. Thinking back to our friend from last month, if he also had a poor memory you might want to write the instructions down on a sheet of paper with "how to walk in a square" at the top.

Whenever he wanted to walk in a square he could follow the instructions without you having to be there to tell him how to do it. It's just the same with Logo and a procedure is like the sheet of paper. We'll call ours **box**.

First type:

```
ts
```

which expands the text window to cover the whole screen. This is more convenient for writing procedure.

Now type:

```
to box
```

and press Return/Enter. Notice how the? prompt has changed into>. This means that Logo recognises we are defining a procedure and want to store the commands rather than execute them straight away.

Enter the list of instructions we used last time:

```
repeat 4 [fd 1ØØ rt 9Ø]
```

and finish off with:

```
end
```

This means you have completed the definition of **box** and Logo will tell you so with the message "box defined".

Logo is an extensible language, meaning that when you define a procedure such as **box** you have added a new command to the language. Instructions that are built in, like **fd** and **rt** are know as primitives and from now on **box** will be treated just like any other.

If you clear the screen and enter the name of the procedure:

```
cs
box
```

The interpreter will match up the name with the list of stored instructions and use the turtle to draw a square.

When writing a program it's often handy to have a procedure which draws a box. It can form a border round text and is a useful building block when drawing more complex pictures. The procedure we have just defined is the sort of thing we need but it's a bit restricted. The only type of box it will draw is a square of fixed size.

We could define lots of procedures for different sizes and shapes of box but that would be terribly long–winded. As you might expect with such a friendly language, Logo has a special feature to help us out. We can define **box** in such a way that we can call it to draw a rectangle and say how big we want the sides to be.

Here's how it's done:

```
to box :hs :vs
repeat 2 [ fd : vs rt 9Ø fd : hs
rt 9Ø ]
end
```

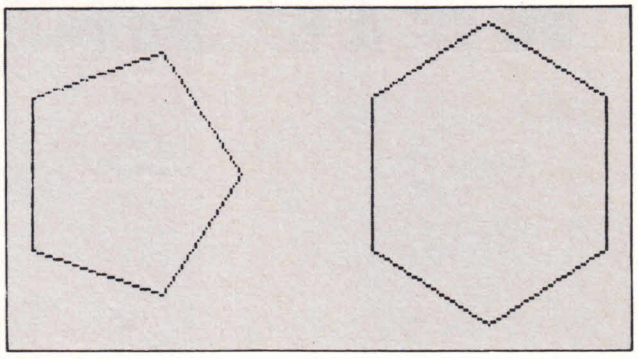


Figure I: Can you produce these shapes?

The **:hs** and **:vs** on the first line of box are the names of variables. When used with a colon on the first line of the definition they will tell Logo that whenever box is used two numbers should follow that say how big the horizontal and vertical sides are going to be. Logo puts the numbers in the variables hs and vs.

You can test it with:

```
cs

box 5Ø 1ØØ
```

which puts the numbers 50 and 100 in the variables.

For those of you who are new to computing, let alone Logo, variables are used in some form in all computer languages and if you can remember algebra from school you probably understand them already.

A variable is a storage compartment, referred to by a name which you choose, used to hold numbers or letters which can be retrieved later in a program.

When the second line of the procedure is executed the **repeat 2** statement says that whatever is inside the following square brackets has to be done twice. The first instruction is to send the turtle forward, but instead of the definite number of units we've been using so far we see the variable name vs preceded by a colon.

Remember, putting a colon in front of a variable name outputs the value held in the variable, so Logo sees the expression **:vs** as the value held in vs – in this case 50.

Similarly **:hs** evaluates to the value stored in hs which we have specified as 100. With that in mind you should be able to work out what **box 50 100** does without typing it in, but I suggest that you do just to make sure you were right.

This way of giving a procedure some values to work with is known as passing parameters. We can apply it to other shapes which can then be used to build up more complicated pictures.

For instance to draw an equilateral triangle of any size we can

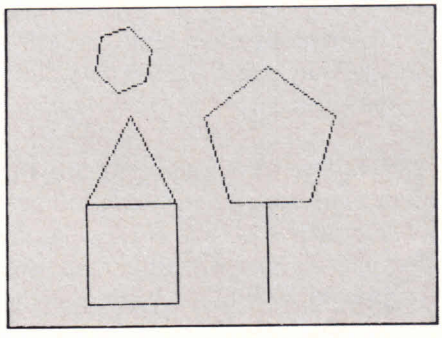


Figure II: More homework

define a procedure **triangle** using:

```
to triangle :sz
repeat 3 [fd :sz rt 12Ø]
end
```

and draw different sizes of triangle with commands like:

```
cs
triangle 5Ø
```

and:

```
cs
triangle 1ØØ
```

The procedures we have defined so far illustrate an important point in turtle graphics. Have you noticed how the turtle always ends up pointing the way it started?

To do this all the angles it turned through must have added up to 360 degrees. So to work out the angle needed in **box** which has four sides we calculate 360/4 which is 90 degrees. A triangle has three sides so 360/3 gives us 120 degrees.

This idea holds true in any situation where you want to turn through an angle in more than one step – divide the total angle by the number of steps.

The shapes in Figure I were drawn with similar procedures and only required the length of a side to be passed as a parameter.

Can you work out how to define and use them to help construct something like the picture in Figure II?

You'll need to use the **pu** and **pd** instructions to move the turtle without drawing lines. Try to imagine a person carrying a large pen and walking about on a big piece of paper. How would you tell him to draw the picture? If you can work that out you are well on the way towards telling the turtle how to do it.

• *Next month we'll develop more useful graphics procedures and start working towards a simple sketch pad program. We'll also start using the save and load commands so that you can keep a permanent record of your procedures.*

This is how to make a working Logo disc. It's always better to work from a copy rather than the master disc in case you accidentally lose or corrupt important files.

Installing CP/M 2.2 Logo on the CPC464+DD1/CPC664

• Put side 1 of your system disc in the drive and type:

iCPM

• When the **A>** prompt appears type:

DISCCOPY

• When prompted for the source disc insert side 2 of your system disc (Logo).

• When asked for the destination disc insert your working Logo disc.

• After the copy is complete reset your machine, put your new Logo disc in the drive and type:

iCPM

• Logo will be loaded automatically.

Installing CP/M + Logo on the CPC6128

• Put side 1 of your CP/M disc in the drive and type:

iCPM

• When the A> prompt appears, type:

DISCKIT3

• When the menu appears, select the format option.

• Select system format, insert your Logo disc and follow the prompts.

• Exit back to CPM and insert side 1 of your master disc.

• Type **PIP** and wait for the * prompt.

• Type:

```
b:a:*.ems
b:a:submit.com
```

• When you are asked for the disc for B, put in your Logo disc. Disc for A refers to the disc you are copying from.

• When you are asked for the disc for B, put in your Logo disc. Disc for A refers to the disc you are copying from.

• Insert side 3 of your master disc (DR.LOGO & HELP).

• Type the following, inserting your working Logo disc when asked for Disc for B.

```
b:=a:setkeys.com
b:=a:logo3.com
b:=a:keys.drl
b:profile.sub=a:a:logo3.sub
```

• Reset the machine. Put your working Logo disc in the drive and CAT it. You should see the following files:

```
C10CPM3.EMS LOGO.COM
SETKEYS.COM KEYS.DRL
PROFILE.SUB SUBMIT.COM
```

• *Type **iCPM** and Logo will be loaded automatically.*

DECISIONS

By
PETE BIBBY

This month we'll go into IF a little more deeply, seeing how it can be used with strings and joint comparisons.

Have you noticed how relentless the Amstrad is? It starts at the beginning of a program and works through it line by line, obeying each and every line. A FOR ... NEXT or WHILE ... WEND loop might send it round the houses but each line in a program is obeyed at least once.

However sometimes you don't want this to happen. It might be that you only want a line to be obeyed if a certain condition is true. A bank manager might want his computer to tell him if a customer is overdrawn. But he only wants the line:

```
PRINT account$; "is in the red"
```

obeyed when account$ is in the red. If not then that line is not to be obeyed.

Locomotive Basic allows for this type of thing by having the IF .... THEN statement. It works along the lines of:

```
IF condition is true THEN do something
```

evaluating the condition and if – and only if – the condition is found to be true, obeying the rest of the line after the THEN. If the condition isn't true then all the code after the THEN is ignored.

Figure I shows the comparisons that can be used and Program I shows the first of them in action.

Operator	Meaning
=	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
<>	not equal to

Figure I: Comparative operators

```
10 REM Program I
20 FOR loop=1 TO 10
30 READ number
40 IF number=5 THEN PRINT
    number "is number"
50 NEXT loop
60 DATA 0,1,2,3,4
70 DATA 5,6,7,8,9
```

Program I

The comparison's made in line 40. Here the contents of a variable, *number*, are compared with the number 5. It's easy to see that if *number* holds 5 then the condition is true (5=5) and the rest of the line is obeyed.

Notice that as the FOR ... NEXT loop is obeyed, 10 numbers are read into *number*. However it's only when the condition is true (when *number*=5) that the message is printed. The line is only obeyed when conditions are right.

Following close on the compara-

tive operator = (as such things are known), is <>. This is the opposite of = and means, unsurprisingly, "is not equal to". Try changing line 40 of Program I to:

```
4Ø IF number <> 5 THEN PRINT
    number;"is not five"
```

and you'll see what it does. Now the condition is true whenever *number* is not equal to 5. When this is the case the message is displayed. When *number* does hold 5 then the condition can't be true (how can you have 5 not equal to 5?) and the rest of the line is ignored. So you get a message for every value of *number* but 5.

The next operator we'll meet is < which means "less than". Again, alter the line in Program I, this time to:

```
4Ø IF number < 5 THEN PRINT num
    ber;"
```

is less than 5"

and see what happens. Now the rest of the line after the THEN is only obeyed when *number* holds a value less than 5. The result is that the program gives messages for values of 0 to 4 none after that.

Don't get the "less than" operator <, confused with the "less than or equal to" operator <=. They're subtly different in their effect. <= also allows the case where *number* is equal to as well as less than 5. If you try:

```
4Ø If number <= 5 THEN PRINT num
    ber;" is either less than 5 or equal
    to 5"
```

in Program I you'll see that now you get the numbers 0 to 5 displayed.

Since we've had a "less than" operator it seems logical that there should be a "greater than" operator. There is – it's >. Using:

```
4Ø IF number > 5 THEN PRINT num
    ber;" is greater than 5"
```

will give you 6,7,8,9 as your reward, all the values of number that are greater than 5.

Not surprisingly there's also "greater than or equal to" operator. It's >=. A quick alteration to Program I in the form of:

```
4Ø IF number >= 5 THEN PRINT
    number;"is either greater than 5 or
    equal to 5"
```

will show you what it does. Now the output is 5,6,7,8,9. The "equal to" part of the condition has accepted the case where *number* is 5.

Try altering the comparisons made in line 40, using other values than 5. Also, you can change the numbers in the data lines and see what happens. Be sure to note the difference between < and <=,> and>=. It may seem small but can be a potent force for program, faults.

Notice that the operators are in opposites. If something is <= to something else, it can't be > than it. That is, if number is less than or equal to 5, say 4, then it obviously isn't greater than 5. Similarly if it's greater than or equal to 5, say 6, then it can't be less than 5. The conditions are mutually exclusive.

Don't worry if this seems a bit academic – when you come to use IF ... THEN to get your programs to take decisions, you'll understand.

```
1Ø REM Program II
2Ø PRINT "Give me a number"
3Ø INPUT number
4Ø If number<5 GOTO 7Ø
5Ø PRINT number "is greater than or
    equal to 5"
6Ø GOTO 8Ø
7Ø PRINT number "is less than 5"
8Ø END
```

Program II

In the above program we've just compared the value in a variable with a number. We could also compare two variables or an expression such as:

```
IF oneVariable > twoVariable THEN ...
    IF 2*one Variable < 3*twoVariable
    THEN ...
```

Also the examples chosen have used the IF ... THEN conditional statement with just a PRINT after the THEN. There can in fact be all kinds of keywords after the THEN such as LET or CLS.

You can even send the program hurtling all over the place. This is done using that black sheep of the keyword family. GOTO. If you use GOTO with an IF, you don't need to use the THEN, though the Amstrad won't be upset if you do. Program II shows IF ... GOTO in action.

Notice the horrible way you have to have another GOTO in line 60 to stop things clashing. Try leaving it out and see what happens.

The END of line 80 just stops the micro going any further and gives line 60's GOTO somewhere to aim.

Whichever way they are used, IF statements are extremely important. They allow our programs to make choices, sometimes performing a line, sometimes not. This

makes them much more flexible and so more useful.

Now that we're *familiar* with IFs and conditions again, let's go deeper. Often we don't want to test for just one condition, we want to test for two. For example, looking at Program I again, we may want a number that lies between 4 and 7. Here there are two conditions to be considered. We want *number* to be greater than 3 and at the same time *number* has to be less than 8. If you use:

```
40 IF number>3 AND number <8 THEN
   PRINT number
```

you'll see how this is done. All we've done is to join our two conditions with a logical operator, AND. Now both conditions have to be true before the code after the THEN is obeyed. The result is that 4, 5, 6 and 7 are returned.

Try the following line:

```
40 IF number>=3 AND number <=8
   THEN PRINT number
```

Can you see how both conditions combine to give 3, 4, 5, 6, 7 and 8?

While we often want two conditions both to be true before we go on to the bit after the THEN, sometimes we want the line to be obeyed if one or other or both of two conditions are true. We may want to go out if it's sunny or dry or both.

Here only one of the conditions has to be true for us to be out the door. The only way we stay in is if both conditions are false, that is, if it's both wet and, at the same time, dark.

There's a logical operator to deal with this, the aptly named OR. You can see it in action with:

```
40 IF number <3 OR number >7 THEN
   PRINT number
```

Here the line is obeyed if *number* is less than 3 or greater than 7. The result is that 0, 1, 2, 8 and 9 appear.

Can you explain why:

```
40 IF number <=3 OR number >=7
   THEN PRINT number
```

gives, 0, 1, 2, 3, 7, 8 and 9? The addition of the equals sign means that 3 and 7 come in from the cold.

As with single comparisons, try changing Program I to try different combinations of conditions acting on different numbers. And try to predict what the outcome will be before you run the program.

It's all too easy to use <= instead of <, or AND for OR, allowing unexpected values to slip through the condition. In small programs like Program I this is no problem but in more realistic programs all hell can break loose. And it can be difficult to trace the mistake, so be warned.

As well as AND and OR you can use another logical operator with the conditions of an IF. This is the NOT operator which negates the condition. It's easier to use in practice than it is to explain, though at times it needs care. If you change line 40 of our long–suffering Program I to:

```
IF NOT number=5 THEN PRINT num
   ber
```

you'll see how the output is the reverse of the previous. This is to be expected as the condition is now true when *number* is NOT 5.

Hence the part after the THEN is obeyed when *number* is 0, 1 and so on. In fact every time except when it is 5. Try using NOT on some of the earlier conditions we tested in Program I and see what effect it has on the output. You'll see that it reverses it.

But beware when you come to dealing with joint conditions. See if you can tell the difference between:

```
40 IF NOT (number >=3 AND number
   <=8) THEN PRINT number
```

and:

```
40 IF NOT number >=3 AND number
   <=8 THEN PRINT number
```

The one gives 0, 1, 2, 9, the second just 0, 1 and 2. In the earlier case the brackets around the two ANDed conditions ensure that the NOT applies to them both.

In the second, the NOT only applies to the first condition. The second one is left unNOTted (if there's such a word!) and so won't allow the 9 to be printed because it's greater than 8.

The difference is subtle but can be important. I avoid NOTs like the plague and use another pair of conditions to achieve the same effect.

Just as we can compare numbers and numeric variables, so we can compare strings. This is possible because, as you know, computers work with numbers. Micros store strings as numbers and when we ask our Amstrad to compare, say A with B it compares the numbers it holds for the strings. These numbers are the Ascii codes we met in the September 1985 issue.

Happily, we don't have to be concerned with the numbers, we can just use the strings with the com-

Continued on page 24

TAKE A PEEK, double it ... and make a DEEK

On numerous occasions we need to peek into the Amstrad's memory, for instance to examine certain values in the system variables area.

Values less than 256 can be stored away in one physical memory location – a byte. An example of this is the TAB size which we can alter using the command ZONE. Its default value is 13 and this number can be found at &AE79 on the CPC464.

Most of the system variables occupy two or more memory locations thus allowing larger values to be stored such as table addresses, pointers and so on.

It is common practice for computers based on the Z80 to store these 16 bit numbers in a reversed fashion. The first byte holds the least significant part of the number and the most significant part moves into the second byte. The following command retrieves that two byte value:

PATRICK de GEEST introduces the DEEK command, and shows how it creates access to memory

```
PRINT PEEK(first_byte) + 256
* PEEK(second_byte)
```

For instance the highest memory location available to Basic, HIMEM, is stored at locations &AE7B and &AE7C. It can be found by providing the above command with the right address:

```
PRINT PEEK(&AE7B)+256*(&AE7C)
```

giving 42619 on my micro. Peeking two consecutive bytes in ram or rom in the fashion I've just described is more commonly named a double peek.

Some Basic implementations have a double peek already in their vocabulary list often abbreviated to DEEK. A deek into memory always returns a value between 0 and 65535, which is the maximum number that can be stored in two memory bytes.

Two bytes indeed allow for 65536 different bit configurations. It is this missing command DEEK that we are going to simulate using the RSX extension facility which is available via machine code.

One advantage is certainly the ease with which we can have access to the memory and at the same time assign a variable with the double peeked value, We can design the RSX syntax as follows:

```
num%=Ø
|DEEK, address,@num%
```

Its function is to double peek into *address* and *address* + 1 and to store the value in the integer variable *num%*. The exact location of this variable is passed as a parameter using @

The variable *num%* must be assigned a value at least once somewhere in the program otherwise Basic isn't aware of its existence and won't be able to pass its exact location in memory as a parameter. Basic informs you of this eventuality with the error message "Improper argument".

Integer variables like *num%* are convenient since these also take just two bytes to store their values.

Note that all parameters passed to RSXs are converted into two byte values and if necessary Basic rounds numbers off. If pi were to be passed our machine code routine would pick up the number 3 in the low byte and 0 in the high byte.

But how does our routine know where to look for these parameter values?

This is very simple. During the interpretation by Basic of the RSX command the register pair IX will be initialised with the memory address of a parameter block. While Basic digests these parameters from left to right it pushes the values on the stack. As a consequence the first item in this table will be the last parameter in the RSX command.

Using this index register makes it easy to pickup the parameter values. The logic behind it can be visualised as follows:

```
IDEEK, (IX+n) (IX+(n–1))...(IX+Ø)
```

```
org &AØØØ
ld bc, Table
ld hl, Buffer
jp &BCD1 ;log RSX
Table: defw Syntax
jp RSX
Buffer: defs 4
Syntax: defm "DEE"
defb "K"+&8Ø
defb &ØØ
RSX: cp2 ; 2 parameters?
ret nz
ld h,(ix+3) ;hl=address
ld l,(ix+2)
ld d,(ix+1) ; de=num% addr
ld e,(ix+Ø)
ld a,(hl) ; peek
ld (de),a ; store
inc de
inc hl
ld a,(hl) ; peek
ld (de),a ; store
ret ; end
```

Listing I

```
1Ø REM PROGRAM 1
2Ø MEMORY &9FFF:count =45
3Ø FOR x=1 TO count
4Ø READ b$
5Ø POKE &9FFF+x,VAL ("&"+b$)
6Ø NEXT x
7Ø CALL &AØØØ
8Ø END
9Ø REM *** RSX set up ***
1ØØ DATA Ø1,Ø9,AØ,21,ØE,
     AØ,C3,D1,BC,12,AØ,C3
11Ø DATA 17,AØ,ØØ,ØØ,ØØ,
     ØØ,44,45,45,CB,ØØ
12Ø REM *** Machine code ***
13Ø DATA FE,Ø2,CØ,DD,66,
     Ø3,DD,6E,Ø2,DD,56,Ø1
14Ø DATA DD,5E,ØØ,7E,12,
     13,23,7E,12,C9
15Ø REM ————————
16Ø use: IDEEK, address,
     @num%
17Ø REM ————————
```

Program I

Here's how to set up the IDEEK command in machine mode:

The first part of the program sets up the RSX and the second part is our machine code routine itself.

To introduce this RSX to the firmware we execute a CALL &AOOO just once. From the moment the ready prompt reappears we have the facility to use IDEEK.

Those without an assembler should enter and run Program I.

Now it is time to experiment. Let's poke a number into memory say 5156, and see if we can double peek it back. Separated into low and high byte we have 5156=36 + 256 x 20 so enter:

```
POKE &A1Ø1,2Ø
POKE &A1ØØ,36
```

And now the final test:

```
Num%=Ø
IDEEK,&A1ØØ,@num%
PRINT num%
```

which gives 5156.

Now let's try a larger number for instance 55156, 116 + 215 x 256:

```
POKE &A1ØØ,116
POKE &A1Ø1,215
IDEEK, &A1ØØ, @num%
PRINT num%
```

This gives –10380 instead of the expected 551561.

The cause has absolutely nothing to do with our RSX routine. Falling back on the normal deeking method doesn't help solve the problem either as a quick verification proves:

```
num%=PEEK(&A1ØØ)+256*
    PEEK(&A1Ø1)
PRINT num%
```

The result in this case is not an erroneous number but an "Overflow" error message.

The reason behind these anomalies lies in the fact that Locomotive Basic treats its integers as signed integers whereas we expected them to be unsigned. What is the difference?

In two bytes the total number of possible bit configurations is 65536 and when dealing with absolute unsigned integers we consider the range 0 to 65535 to be totally positive.

However, when speaking of signed integers as Basic does, this range of 65536 integers starts not at 0 but at – 32768 and ranges up to 32767. Advanced machine code programmers should immediately recognise the analogy with two's complement numbers.

The range of numbers is shifted halfway down the scale so we can work with negative numbers as well.

As a consequence all numbers greater than + 32767 and all numbers smaller than – 32768 are forced to be of type real. But by appending a % sign after the variable name *num*. Basic was prohibited from making the conversion into a real number and all it could do was print the error message.

If the % sign is left off then Basic can execute the command:

```
num=PEEK (&A1ØØ) +
    256*PEEK(&A1Ø1)
PRINT num
```

When this statement gives values above + 32767 we know for sure that the variable *num* will be a real since to handle larger numbers Basic is obliged to leave the integer framework.

It stems from the fact that using variables of the default type Basic automatically switches to real number notation once the integer limits become insufficient.

The main advantage of integer notation is that these numbers are readily available to be manipulated by Basic. No preliminary conversions are needed and everything is very quickly processed.

That cannot be said when working with reals. Performing operations with reals involves much more machine code compared with integer operations and as a result the execution time will significantly increase.

Also important is that when the variable has a ! sign to it, then even when the value can be written as an integer Basic nevertheless resorts to reals. To see this demonstrated enter:

```
POKE &A1ØØ,36
POKE &A1Ø1,2Ø
num!=PEEK(&A1ØØ)+256*
    PEEK(&A1Ø1)
PRINT num!

POKE &A1ØØ,116
POKE &A1Ø1,215
num!=PEEK(&A1ØØ)+256*PEEK
    (&A1Ø1)
PRINT num!
```

The first number is again 5156 but this time we know its internal format is a real. The second number now is 55156 as required and it is no more the signed integer equivalent (–10380) and is also a real.

• Next month we'll take a closer look at the way reals are stored and we'll develop our simple DEEK routines further.

Machine Code

continued from page 10

```
3OOC   3E ØA      LD A,&ØA
3ØØE   CD 5A BB   CALL CharOut
3Ø11   3E ØD      LD A,&ØD
3Ø13   CD BB 5A   CALL CharOut
```

are just to print out a line feed followed by a carriage return, ensuring our lines are separate. Try leaving either or both out and see what happens. (Remember that your jumps will have changed.)

Right, that's plenty for one month. I'll leave you with two things to ponder, though.

If we're simply obtaining the value of B from C, why bother with C at all? Actually, that's easy, but it's the sort of daft thinking that can occur to you after your mind's been numbed by several hours of machine code programming.

My final poser is a little more demanding. Try turning the triangle upside down, with one asterisk at the top, and eight at the bottom.

```
CP   A   BF
CP   B   B8
CP   C   B9
CP   D   BA
CP   E   BB
CP   H   BC
CP   L   BD
```

Table I: CP r

First Steps

Continued from page 21

And we can think of the < operator as meaning "comes before" and > as meaning comes after. So comparing strings you should see that:

```
a>1
a>A
a<z
a>Z
1<9
<A
ab<ac
aa>aA
bc>bcd
12>1234
```

are all true.

You'll notice that some of the above are more than one character long. To compare two strings the Amstrad looks at the first character. If these are the same then it goes on to the next character, if any.

Should there not be a next character, then the shorter string is deemed the one that comes first. Hence BAR comes before BARD. If you think about it, this is the way things are done in dictionaries and other alphabetical lists.

Program III not only shows you string comparisons in

```
1Ø REM Program III
2Ø PRINT "Enter two strings"
3Ø INPUT first$,second$
4Ø IF first$<second$ THEN PRINT first
    $;"comes before";second$
6Ø IF first$>second$ THEN PRINT first
    $;"comes after";second$
```

Program III

action, it also lets you explore the way the Amstrad orders strings. Play around with it until you've grasped how it does it. And then, if you're really keen to test what you've learnt, try writing a program that will take five strings and display them in alphabetical order.

• *That should keep you busy until next month when we see what ELSE can come with IF.*

Christmas with the Amstrad...

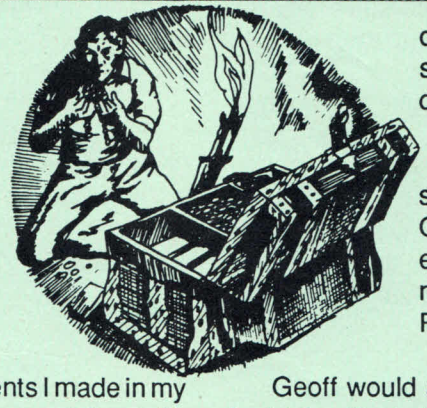
The offices of Computing With the Amstrad will be closed for the Christmas break from 23 December and re-open on 11 January 1988 ... soooo, Happy Christmas to all our dear readers, and enjoy our final issue for 1987as you laze in the sun ...

Adventure

Go adventuring with Gandalf

MORE TO MORDOR

Since reviewing Shadows of Mordor, the sequel to Lord of the Rings, I have received a copy of the playing hints that sometimes – though not often enough – come with a review copy of a game.



Reading these shows where I went wrong and also the degree of character interaction needed to complete the game.

I am going to alter the marks or comments I made in my review but would point out that in the documentation with the sheet, it says that there are limited graphics in the game.

This is something I didn't see and didn't know about when I played it.

Graham Wheeler tells me that two adventures I hadn't heard of, Theseus and Nythyel, were written by Tony Collins of the 50/50 club.

This is a business where ideas are sent to be coded into adventures, marketed, and the proceeds split equally between the programmer and the sender of the original idea. This is a splendid idea and I would like Tony to send in some of these games for review together with more information about the club.

Glynn White, the winner of last year's competition to provide the most maps and solutions, has sent in yet more material. I shall be publishing some of it over the coming months and would like to thank him for all his work.

Finally, before we look at this month's problems, I'd like to apologise to readers who have written and haven't had a reply.

I have been moving house and several answers to letters have been delayed. It's not easy to solve problems when you don't remember which packing case they're in.

Geoff Parker is in difficulties with Leather Goddesses of Phobos. To get anywhere with King Mitre and his daughter, you need to take the salesman's machine and the untangling cream to the throne room.

Put the cream in the machine and start it, to make the untangling cream. Cover the princess with it. Buying an exit? Get the penguins to change the 10 marsmid coin. Moving in the catacombs? Read page 7 of the comic book.

Geoff would also like to know where he can get the Invisiclue Books for Infocom games. I'm sorry I can't help. Has any reader seen them?

Barry Spiers is stuck in three adventures. In Heroes of Karn, he would like to know how to get the jade flower; where to find Haldir; how to get up the broken stairs in the plant room and how to kill the bat.

The answers in order are – drop the pillow and play the flute; in the crypt; water the plant; tell Beren to kill the bat with the falcon.

In Terrormolinos he wants to know how to save Ken at the bullfight; and in Snowball, how to get through the trapdoor.

Wave an unknotted hanky and take the bull to the china shop. There are several trapdoors in Snowball. I'm going to assume you are talking of the one in the initial location. In this case climb the coffin and go up.

YOUR PROBLEMS ANSWERED

David Roberts and Jonathan Hutchings are stuck at the beginning of Seabase Delta. They have got the documents and travel pass and called the travel car but cannot board.

You must enter the car, wear the belt and insert the card and the car will travel to the next station.

They have also asked me to publish a map of this adventure so part one appears this month. Remember, if there are any maps you want to see please let me know.

Graham Wheeler has written to help solve several problems that have cropped up in previous months.

In Spytrek keep giving money to the tramp at the airport until you get a key and an umbrella and then go to the Eiffel Tower.

Here you can unlock the door which will automatically light the staircase. Walk up and down the stairs until you have lost enough weight to be able to float away in the balloon.

Ian Pullen has a problem with Castle Blackstar and Graham writes: You have to be beside the weather vane on the top of the castle. There is a secret passage leading up from the other side of the mirror in the Duke's bedroom – and you need the orb and the broomstick when you try to fly. Saying 'Abracadabra', going up and touching the stars will transport you to a passage in the realm of Artemis.

COMPLAINTS DEPT

Last month I said that some of the smaller software houses seemed to feel that reviewers are biased in favour of glossier packages produced by larger companies.

Personally I don't believe this and, it's not true in my case.

Because of this complaint, I'm starting a new section in my column. If anyone has any comments or complaints write in and, where space permits, I will print them.

READERS' HALL OF FAME

Message from Andromeda – A solution by Anthony Robinson

Read the message and reply. When the ship lands, open the airlock, leave the ship and close the airlock. Go to the workshop, take the knife, then go to the antechamber and put on the gloves.

Now go to the red room for the rod, and the control room for the detonator. In the mirrored room, point the rod at the plate and drop it. You can now go South and rotate and take the sphere. In the blue room, drop the sphere on the pedestal and rotate it.

Objects	
Key	Use to unlock the wooden door
Large stone	Has a trapdoor underneath
Match	Strike, then throw at gun powder
Plank	Put it across the pit
Rod	Put it in the hole
Sextant	A red herring
Shoe	A red herring
Skull	A red herring
Spear	Throw this at the octopus
Watch	Give this to the cannibals

Visit the store room for the explosives and place them in the eastern end of the cavern. Go to the western end of the cavern and press the detonator button then drop it.

Get the space axe from the hall of the ancients and the fungus from the plateau before going to the sloping corridor where you throw the fungus at the slug.

In the torture chamber, cut the ropes, drop the knife and take the stones. At the crystal bridge, throw the stones at the creature then go to the commander's chamber and kill the commander with crocodile.

The Neverending Story — A solution by Pat Winstanley

Part One

From the clearing go NE, E, Look, take Auryn, E,E,E, take Artax, N,W,NW (Artax dies in the swamps of sadness), NW,S,S,S,E,E,E,SW, take food, eat food, take leather, NE,W,W,W,S,SW,W,N, take branch, S,E,N,NE,W,SW, light branch.

Now go quickly to the foothills before the branch burns out, NE,E,E,E,E,E,E, down, light bushes, down, take box, N,SW,W,N,W,W,W,SE,W,SW,N, take stone, S, Take horn, W,SE, blow horn, take Falkor. Then say to Falkor 'Fly South', E, up, down, S,E, drop box, smash box, look, take fragments, S,S, (Wait until eyes blink and then S), (Go N and S again while you are waiting). Follow screen instructions to load part two.

Part Two

You start deep in the forest to the West of Spook city. Go E,N, drop fragment, drop leather, drop horn, drop stone, E, take glow–globe, W,N, take apple, S, drop apple, N,W, take rope, E,E,NE, take book, read book, drop book, W, remove planks, down, SE,W.

Tie rope, down, down, take pouch, take coin, drop pouch, down, up, E,S,E, take tin, W, take knife, N,E,E,N, open tin, drop tin, N, take rusty key, S,SE,SW, down, cut web, drop knife, W,SW,E.

Unlock cell, E, drop key, W,W, drop coin, W,W, drop coin, W, take gold key, E,E,S,N,E, down, up, E,NW, up, S,S, take apple eat apple, take horn, S, blow horn, E, take Auryn, W, take Falkor, say to Falkor 'Fly East'. Follow screen instructions to load part three.

Part three

You start on a floating asteroid west of the ivory tower. To finish the game you need to find your way through a maze to the childlike empress so go; E,N, unlock door, E,E, up, NW, up, W,W,E,S,S,N,W,up,W,E,E,E,E,up,E, say please,E,E and you will meet her.

Final message

"Despite the Destruction of Fantasia everything here seems normal and tranquil. The empress congratulates you on surviving the Nothing and succeeding in your quest to return Auryn to her, but most of all, for making Bastian believe in Fantasia.

Now that someone from the real world believes in it, Fantasia can be restored".

Bastien, Atreyu and Falkor set off to begin rebuilding Fantasia.

CAN YOU HELP?

Simon Druce wants help with Doomdark's Revenge and the Vera Cruz Affair. If any reader would like to make up a hint sheet I will print it.

Simon has asked how to get the coin in the large cavern in Bored of the Rings part two. Unfortunately, my maps and solutions to this game have still not turned up since moving house. Can anyone help?

Debby Howard is going round in circles trying to find her first puzzle in Robin of Sherlock. Can someone help her get started?

S. Saltmarsh wants help with Dun Durach. He has obtained the following scripts; Gods see all, Art in order, Skars a pearl and Rats in vain.

He wants to know what they mean and what he should do with them. He also needs to find out how to open the locked doors in Claw Lane and in the jail. I didn't find this adventure easy and would welcome any reader's help.

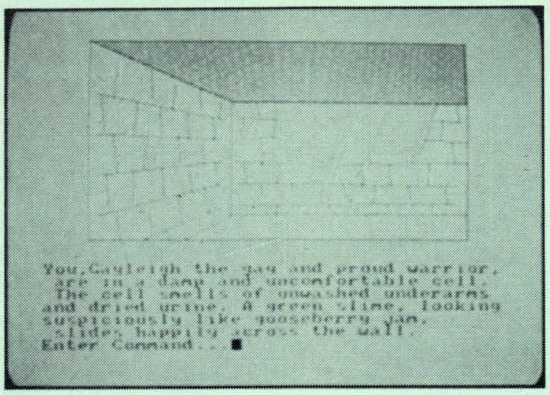
J.Graham is having difficulties with an adventure that I haven't heard of before – The Curse of Sherwood.

He has given the glass and fangs to the witch but keeps sinking in the swamp. He says he has tried every possible exit. Can anyone help him and give me some information about this game?

Dungeons, Amethysts, Alchemists 'n' Everythin'

Atlantis, cassette only

An evil lord has plunged the land into darkness and your task in this GAC'd graphic adventure is to restore the magic amethyst to the alchemist and thereby save the land from certain destruction.



You start in a dungeon, menaced by what seems to be gooseberry jam. Like me you will probably fail at first to realise that movement is possible – but the directions are not displayed, It took me ten minutes of attempting various silly things with the jam before I realised this.

Exploring your surroundings reveals that you are locked in an underground dungeon complex. The door to the outside world is locked.

You soon acquire a loincloth and you may need to cover your modesty with this if you want to meet Gollum. He obviously hasn't yet met Bilbo and you soon obtain another object. Incidentally, try skinny dipping first!

Searching further reveals a key. Unfortunately, it is not the one to the door but will allow you to gain access to an object that, if treated improperly, will lead you to the correct key.

You can now escape to the main body of the adventure. All you have to do now, is find the amethyst and give it to the alchemist.

I love this game and it seems to have everything going for it. The graphics are quite good, complementing the atmosphere, and just the right balance between humour and difficulty has been achieved.

It's not difficult to complete – many of the actions are intended to amuse, not to serve as obstacles. It's been written to entertain not baffle.

As well as puns and cliched situations, the program takes a few gentle swipes at the industry in general. For instance, in the hideout of the Infogong gang, a card is pinned to the wall that reads 'Keep it coming!'

The program is written for all CPC machines and though supplied on tape, is readily transferrable to disc and also features a save game option – two features that make my life a lot easier and are much appreciated.

The only disappointment is that players with roms fitted that take up memory space will have to remove their rom board or the game will not run.

Overall, I find it near impossible to fault this game, and budget priced, it offers marvellous value for money. Highly recommended.

Presentation 90%

Just a simple cassette inlay – but then, what do you expect for the price?

Atmosphere 90%

It's hard to combine humour and atmosphere but they've done it here.

Frustration factor 50%

Not many problems. But if there were more, there would be less humour and the game would be less entertaining.

Value for money 100%

Exceptional.

Overall 90%

Can you afford not to get it?

Passenger on the Wind

Infogrames, cassette only

This is a graphics adventure based on the work of the French cartoonist, Francois Bourgeon and concerns the trials and tribulations of two men and women on the eve of the French Revolution in the 18th century.

Isa, a countess cheated out of her title and lands, meets and falls in love with Hoel, a young Breton sailor. They are captured by the British. Hoel is taken as a prisoner of war and Isa takes a job in Portsmouth as a French tutor.

She befriends Mary, who is having an affair with one of Hoel's guards and the three of them hatch a plan to help Hoel escape. The story begins with this rescue attempt and it is up to your skill to determine the outcome.

The game is not an adventure in the accepted sense. Instead, the screen is split into four main areas each of which fulfils a separate purpose.

The top half is used to display the graphics – one major picture for the 10 episodes that make up the game. As actions are undertaken, small windows open up to depict the scene and the results.

The graphics are very good – though I have seen better.

The lower half is split into three sections depicting: The character currently selected, his or her comments, a list of the choices available and two bars that enable you to change character or choice.

There is no text input – you use the cursor keys or joystick to select the section to update the spacebar to confirm that choice. The only other keys used are for loading or saving the game play.

I found the game very uninteresting – based simply on multiple choice, on much the same lines as the Archers and Adrian Mole. The character set is virtually unreadable and making sense out of what is taking place is itself an adventure.

My main crib is that there is nudity in the program. Although the actual on–screen representation is quite tasteful, I strongly feel that there should be some warning on the packaging. My major objection is the circumstances under which it takes place. One of the women – Mary – is captured and under threat of a knife, is ordered to remove her clothes.

In today's climate, where offences against women are on the increase, I find the sequence to be in bad taste and offensive. In the cinema the package would carry, at the least, a '15' certificate. The situation could have been avoided and certainly isn't justified.

Having climbed down from my soapbox, I have to admit that I can't really find a lot to recommend in this program.

I have never seen Bourgeon's work so I am unable to make comparisons. All I can say is that the graphics are very good.

Presentation 20%

A piece of foam rubber, a small, inadequate booklet and two cassettes.

Atmosphere 60%

Good, because of the high standard of the graphics, If I had been able to read the text my mark might have been higher.

Frustration factor 20%

Very. Mainly because I couldn't read what was happening.

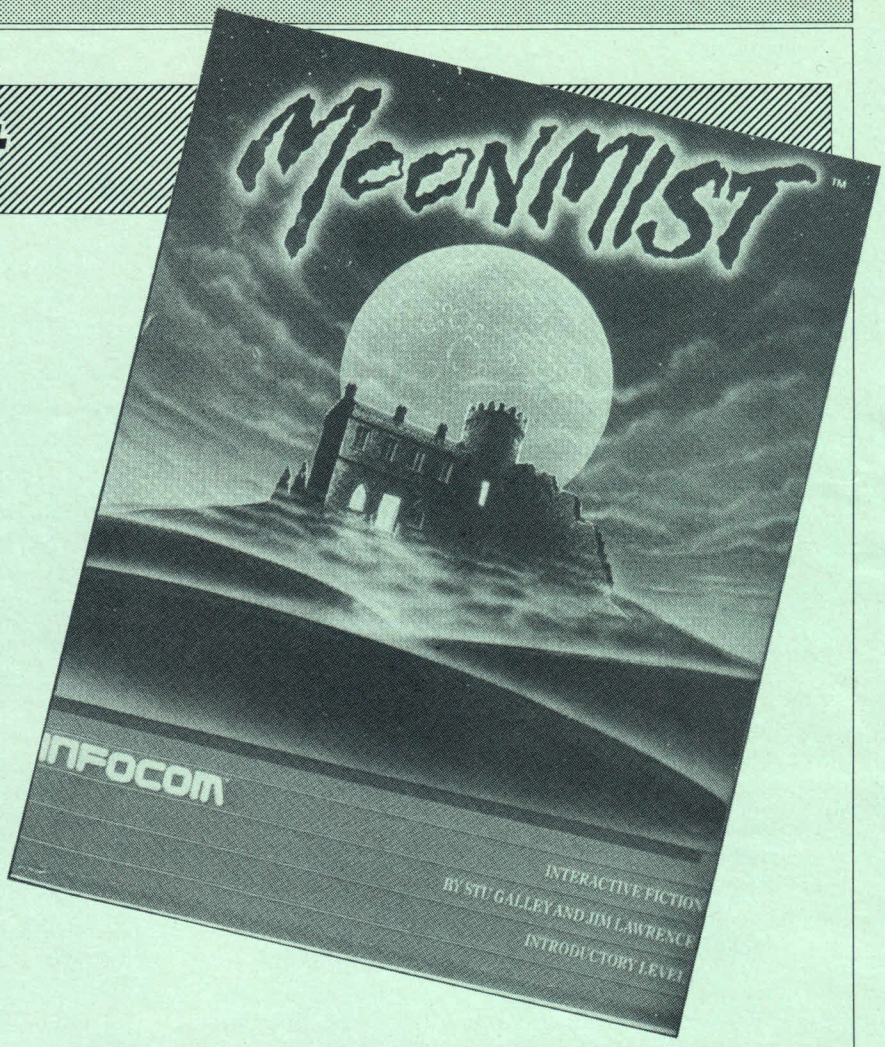
Value for money 10%

Despite there being 10 different episodes, (each of which only takes about 10 minutes to play anyway), it is vastly overpriced.

Overall 30%

Should you buy it? If you like pretty pictures, yes. If you like adventures – this isn't one.

Moonmist



Infocom,disc

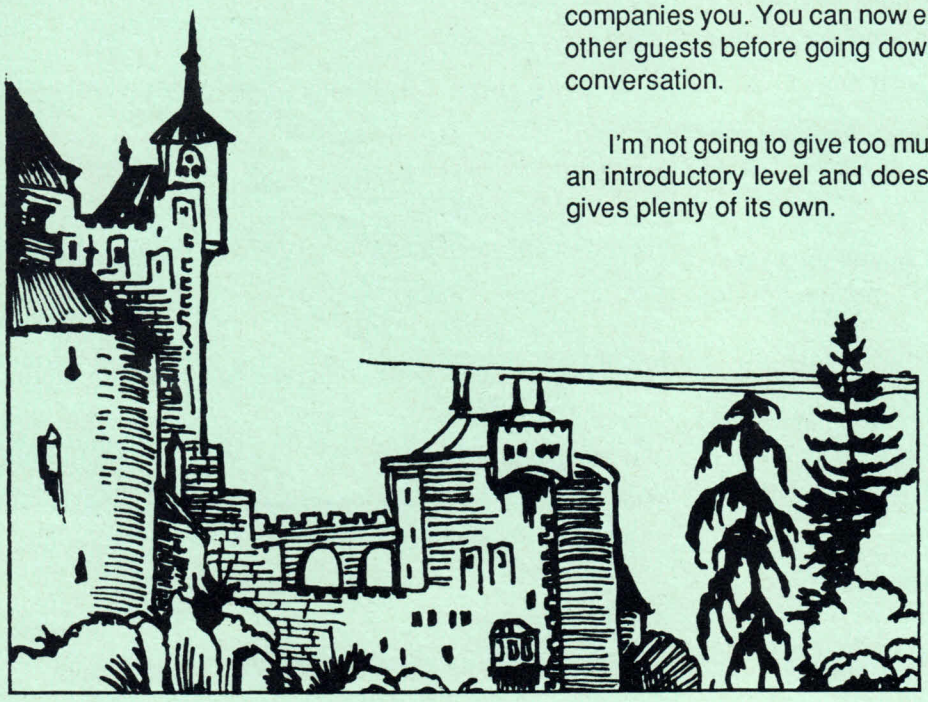
You always thought Tamara was a romantic, and your suspicions are confirmed when you receive a letter from her telling you that she is engaged to marry Lord Jack Tresyllian, the owner of a Cornish castle.

You are amused at first and, of course, very happy for her, but as the letter continues, being the great American detective that you are, your suspicions are aroused. Tamara is very perceptive about people and the house guests seem to be giving her qualms. When she mentions the castle ghost and the fact that Jack's last girlfriend, Deirdre Hallam drowned herself in the castle well you start to worry.

A few days later a second, hysterical letter arrives. Tamara has seen the ghost and it has tried to kill her! Not only that, several people who have seen the ghost recently, have said that it looks like Deirdre. Could the ghost be trying to kill her and could you come to England and discover the truth?

You immediately take a flight and arrive at the castle the same evening. After reassuring Tamara that you have read her letter you are taken in to meet the other house guests. You are then shown to your room and this is an ideal opportunity to question the servant that accompanies you. You can now explore your own room and those of the other guests before going down to dinner and some very interesting conversation.

I'm not going to give too much away about this one. It's classed at an introductory level and doesn't really need any hints from me – it gives plenty of its own.



Your task is to solve the riddles and clues to discover who is trying to kill Tamara and why. There is also a treasure to be found to finish the game.

There are four different variations each with a different culprit, motive and treasure. Your play is dependent on the answer to a question at the start of the game, "What is your favourite colour?" Answering red, blue, yellow or green determines the scenario.

Continues page 35

IREN SOFTWARE

DISCOLOGY NEW

This is the ultimate in disc utilities. Discology consists of 3 programs, a disc editor, a disc explorer and a disc copier.

The Copier
* At the time of going to press Discology really is the most powerful utility ever written for your Amstrad
* Makes full use of all 128K on a 6128.
* Highly intelligent, compresses data allowing whole discs to be copied in one go.
* Full file copier, copy multiple files in one go. Will cope with files of any length.
* Copies files from disc to tape.

The Editor
* Edit any sector, including funny formatted sectors
* Display in Z80 disassembly, basic listing, hex, ASCII, binary, octal & decimal
* Search disc for a given string
* Dump page to printer
* Built in full floating point calculator, hex to decimal conversion etc
* Exceptionally easy to use

The Explorer
* A new concept in disc utilities
* Graphically maps discs and files
* Shows how many sectors on each track and displays on which sectors files are stored.
* Displays full sector information and file information.

"the copier is easily the most powerful for the Amstrad"
"it beats the opposition by doing a better job and having more to offer"
Amstrad Action, Oct 87

Discology makes comprehensive use of pull down menus and is a superb addition to any disc owners software collection. Discology makes extensive use of all 128K on a 6128 and all 64K on a 464. This 100% machine code program offers everything you could dream of plus more.

NEW ★★★ ULTRASOUND ★★★ NEW

The complete sound package for your Amstrad

Ultrasound is a unique suite of 4 programs (plus demos) which will enhance and increase the potential of your computer.

Ultrasound will allow you to digitise about 1 minute of sound without the need for any additional hardware on a 464 (664 & 6128 owners will need a standard cassette recorder) and edit and replay the sound. The sound can be added to your basic or machine code programs.

Synthesoft will turn your CPC keyboard into an electronic synthesisor. Giving you full control over your composition, you can alter the vibrato, octave, sound, and volume. The facility to record and playback your tune is also available.

Soundsoft gives you the facility to quickly and easily create sound effects by directly accessing the sound processor (AY-3-8912) in your computer. These effects can be added to your own programs.

Softtalk will allow you to give your computer a personality. Softtalk will allow your computer to talk to you. Text typed in to the keyboard can be spoken clearly without any additional hardware. Speech can be added to your basic or machine code program easily with the use of new RSX commands.

DISCOVERY PLUS

The ultimate tape to disc transfer program

"Discovery Plus must be the most advanced and probably most efficient tape to tape disc transfer utility to date" Amstrad Action, December 1986.
This program will transfer more games to disc than any other transfer program. The first person who can prove otherwise will receive twice his money back!!
Discovery Plus consists of 4 easy to use programs that together will transfer an extremely high proportion of your software onto disc.
Also includes details on how to transfer over 100 games.
Silver Screwdriver Award Amtix! January 1987.

Discovery Plus now incorporates Splock Trans II

EPROM PROGRAMMER

At last a low cost RELIABLE eprom blower is available for your Amstrad CPC. Contained in a smart case with separate power supply, this unit will enable you to read roms into memory, edit them and blow them onto blank 21v 2764 or 27128 eproms. A ZIF socket allows easy insertion/removal of roms and a through connector allows other add-ons to be attached at the same time. The 100% machine code software (which also run from ROM) allows basic programs to put on and run from a rom and offers full verification, blank checking and a reliable programming option.

CHERRY PAINT NEW

Another new addition to our range, CHERRY PAINT is a superb mode 2 art package. CHERRY PAINT uses icons, pull down menus and windows to provide an easy to use drawing package for your Amstrad.

* Uses 640 * 400 pixels in Mode 2
* Full range of features and options
* Dump designs to your printer in 5 sizes
* Compatible with keyboard, joystick or AMX mouse
* Superb review in Amstrad Action

PRINT MASTER

PRINTMASTER is probably the most useful program that any printer user can buy. JUST LOOK AT ITS COMPREHENSIVE LIST BELOW

* Comes complete with 20 fonts (typefaces)
* Prints any ASCII file (from Tasword/Protext) in a variety of fonts, sizes & styles
* Adds NLQ (Near letter quality) printing to any printer
* Semi proportional spacing available
* Print large posters
* Font designer allows you to create your own fonts
* Dump screens to your printer in 16 shades of grey
* Very easy to use, full instructions and demo's included.

No printer should be without this program!

To order, please use the form on centre page

Greatest ever Savings on Software!!!

We've suspended our usual price list due to the currency fluctuations of the Australian peso but we didn't want you to be disappointed over Christmas so... **Don't miss out on the bargain of this or any year!** We've scooped the pool on over $250,000 worth of software. O.K., we know some of them are oldies - but **they're all goldies! Hurry, some titles are in very short quantity.**

Check these tapes out for value:-

1	HiSoft C	29.99	11	Decision Maker	19.99
2	Devpac Assembler	19.99	12	Invostat	9.99
3	Home Budget	19.99	13	Stockaid	9.99
4	HiSoft Pascal	19.99	14	Tascopy	14.99
5	Screen Designer	19.99	15	Tasprint	14.99
6	Mastercalc	19.99	16	Gamespack II	19.99
7	Masterfile	19.99	17	Halleys Comet	6.99
8	Star Watcher	19.99	18	Qabbalah	6.99
9	Project Planner	19.99	19	Hardball	6.99
10	Entrepreneur	19.99	19A	Frank Bruno's Boxing	6.99

And look at these prices for disk software, *some at less than the cost of a blank disk!*:-

C= all CPC, 6=6128 only,P=PCW only,+=CP/M Plus only (i.e. 6128 & PCW)

C/P-20	Devpac 80	39.99	+-53	C-Basic	39.99
+-21	DR. Draw	39.99	P-54	Catalog	14.99
+-22	DR. Graph	39.99	P-55	Cambase	39.99
+-23	Nevada Fortran	39.99	P-56	Camsoft Invoicing	39.99
P-24	The Knife	14.99	P-57	Brainstorm	34.99
C-25	Devpac Assembler	24.99	C-58	Beach Head	9.99
C-26	HiSoft Pascal	24.99	C-59	Raid	9.99
C-27	Advanced Amsword	29.99	C-60	Grand Prix 2	9.99
C-28	Screen Designer	24.99	C-61	Glen Hoddle Soccer	9.99
C-29	Mastercalc	24.99	C-62	Assault on Port Stanley	9.99
C-30	Project Planner	24.99	C-63	Tank Commander	9.99
C-31	Entrepreneur	24.99	C-64	3D Boxing	9.99
C-32	Decision Maker	24.99	C-65	Cyrus Chess	9.99
C-33	Tascopy	19.99	C-66	Macrocosmica	9.99
P-34	Tasword 8000	29.99	C-67	Doors Of Doom	9.99
P-35	Tasprint 8000	19.99	C-69	Golden Path	9.99
P-36	The Torch	14.99	C-70	Strangeloop	9.99
P-37	Write-Hand-Man	24.99	P-71	Cyrus II - PCW	14.99
6-38	Wordstar Deluxe	59.99	C-72	Hardball	9.99
P-39	Wordstar PCW	69.99	C-73	Frank Bruno's Boxing	9.99
P-40	StarIndex	69.99	C-74	3D Stunt Rider	9.99
P-41	Polyprint	49.99	C-75	Roland In Space	9.99
P-42	Polytype	29.99	C-76	Braxx Bluff	9.99
+-43	Nevada Pilot	29.99	C-77	Alien (The original)	9.99
+-44	DR. MT-Pascal	39.99	C-78	Lords of Midnight	9.99
+-45	NewWord 2	59.99	C-79	Chemistry Revision	29.99
+-46	Nevada Basic	29.99	C-80	Physics Revision	29.99
+-47	Multiplan	59.99	C-81	Sorcery Plus	9.99
+-48	Micro prolog	39.99	C-82	King Solomons Mines Pt. 1	9.99
+-49	Master Planner	59.99	C-83	Fighting Warrior/Exploding Fist	14.99
6-50	Mallard Basic	49.99	P-86	ABC Accounts	149.99
P-51	Mail Merge	34.99	C-87	HiSoft C	39.99

MINIMUM ORDER $25 - C.O.D AVAILABLE - ORDER NOW FOR CHRISTMAS

How to order

1. MAIL ORDER

Please include all information requested on the order form, and make sure you have enclosed your name and address (you'd be surprised!) and the correct amount for the goods you require.

If sending a cheque or ordering using Visa, Mastercard or Bankcard please ensure that the date on your cheque is valid (ie 1987 not 1986) and that your credit card has not expired.

2. TELEPHONE ORDER (008) 030930

Please follow the instructions below carefully ***before*** ringing. This number will only be answered by a machine and cannot be used for general enquiries or messages. Anything other than an order will be ignored – you have been warned!

A) Complete the order form at right as though you were going to order by mail. Do not wait until ringing before deciding which titles you require or trying to find your credit card. The answering machine is voice activated and any pause over a couple of seconds will result in the machine hanging up on you.

B) When the machine answers, read the order from your order form slowly, clearly and distinctly giving all the information you have written down. Where possible leave a telephone number just in case we can't understand or hear your order.

C) This service will be in operation 24 hours a day, every day of the year.

D) Allow 28 days for the delivery of your order – orders which we cannot despatch within 2–3 days of receipt will be advised of the likely delay by mail. Back issue orders will be mailed at the same time as the current issue.

MAIL ORDERS TO: STRATEGY SOFTWARE
PO BOX 11, BLACKMANS BAY, TASMANIA 7052
Enquiries (002) 29 4377
Orders – local call fee – (008) 030930
Use your VISA, MASTERCARD or BANKCARD for mail orders.

ORDER FORM

CAT #	TITLE	PRICE
	TOTAL	

Bankcard ☐ Mastercard ☐ Visa ☐ Cheque ☐

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signed ..... Expiry ☐☐☐☐

COMPUTER

Name ..... Model .....

Address .....

..... Postcode .....

Phone ..... Date .....

Please read *How to order* carefully before using our 008 toll-free phone number

Special Christmas Offer!

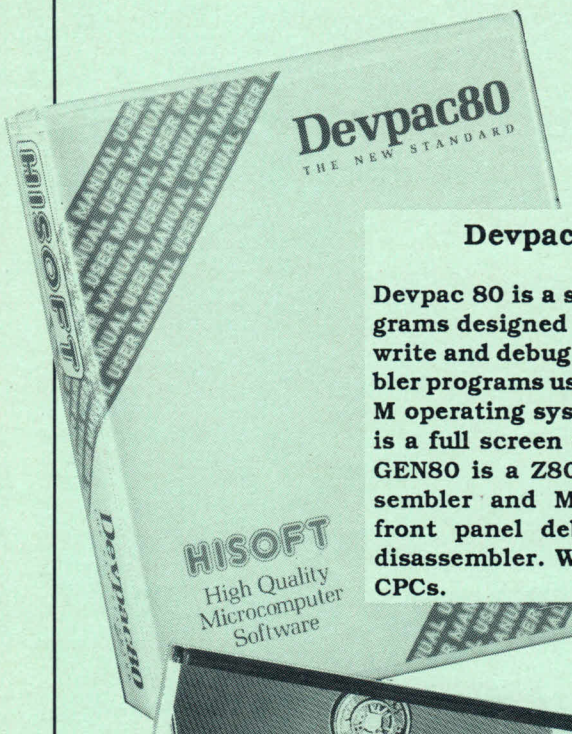
Back issues only $2.40 each for a complete set

$AVINGS ON MAGAZINE$

Use the ORDER FORM above. Just write which issues you want, starting August 86, at only $3.45 each, or get the super bargain of a complete set of 15 for only $36. All prices include pack and post costs.

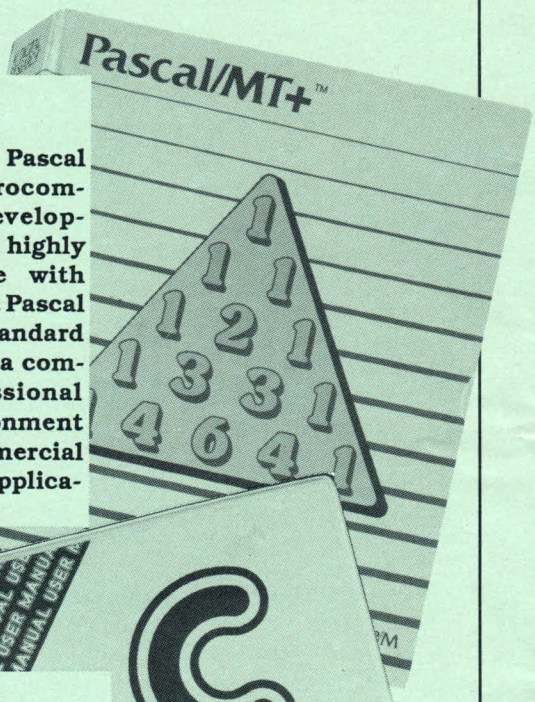
ONLY WHILE STOCKS LAST

SPECIALS FOR SERIOUS USERS



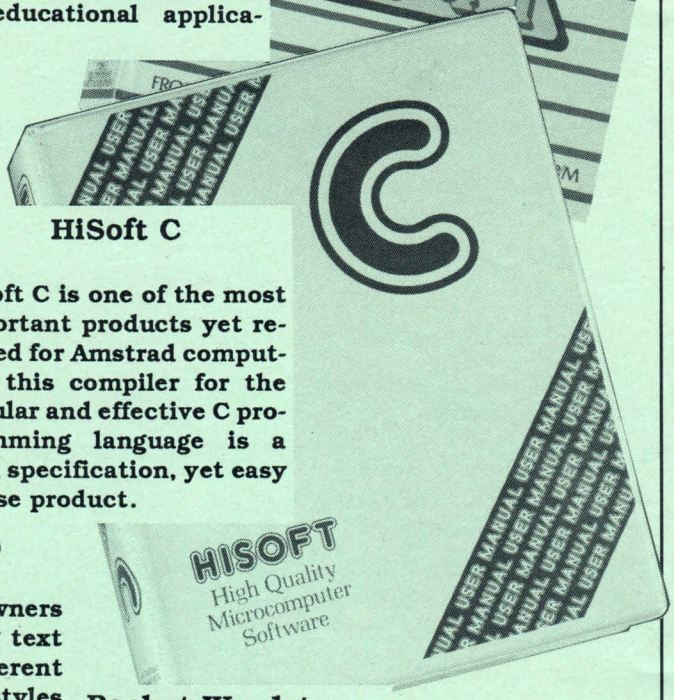
Devpac 80

Devpac 80 is a suite of programs designed to help you write and debug Z80 assembler programs using the CP/M operating system. ED 80 is a full screen text editor, GEN80 is a Z80 macro assembler and MON80 is a front panel debugger and disassembler. Works on all CPCs.



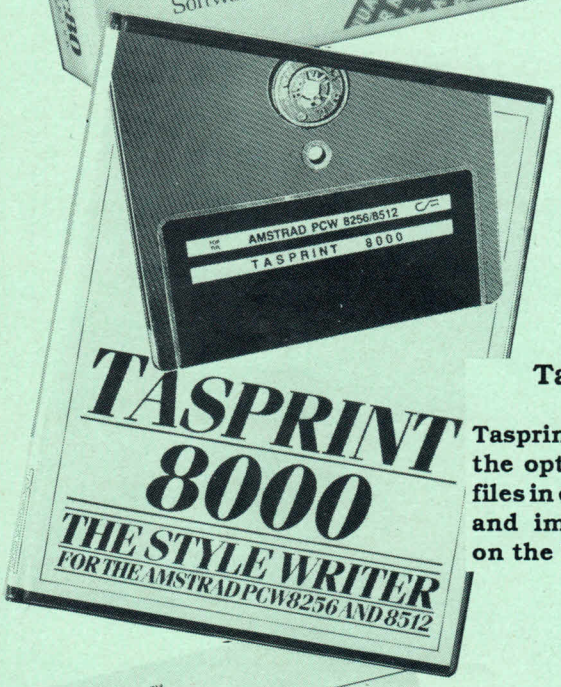
Pascal/MT

The most complete Pascal available for microcomputer program development; Pascal MT is a highly structured language with extensive data types. Pascal MT is the full ISO standard extended to provide a comprehensive, professional programming environment for industrial, commercial and educational applications.



HiSoft C

HiSoft C is one of the most important products yet released for Amstrad computers; this compiler for the popular and effective C programming language is a high specification, yet easy to use product.

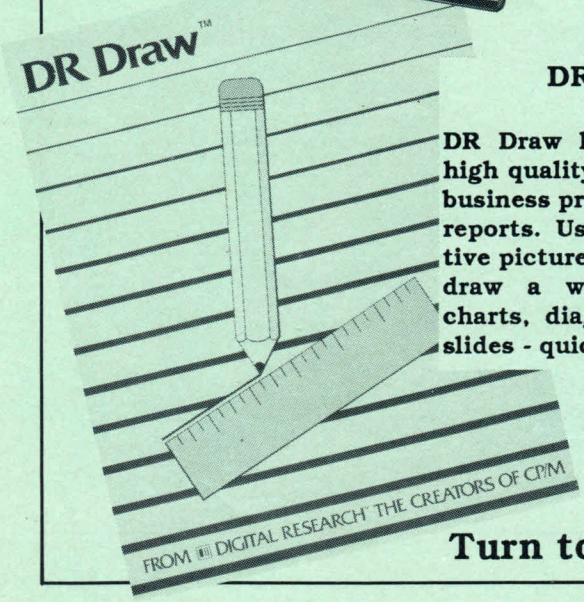


TasPrint 8000

Tasprint gives PCW owners the option of printing text files in one of eight different and impressive typestyles on the PCW printer

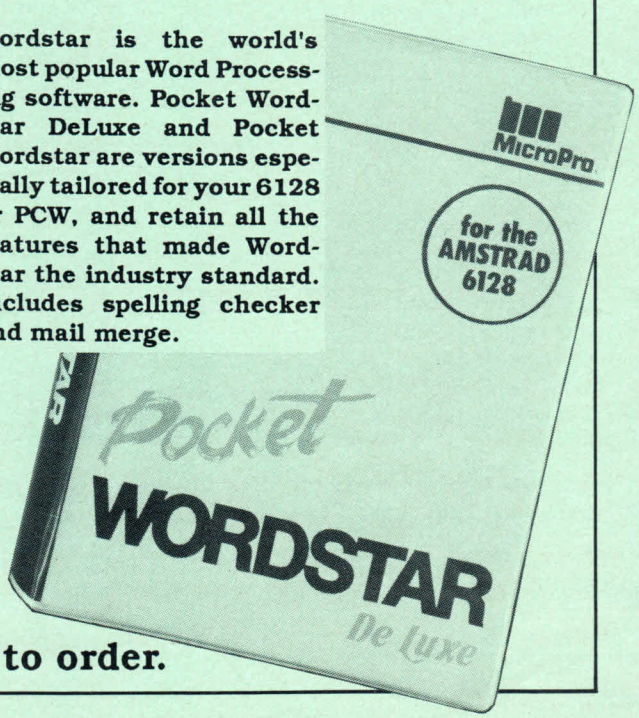
Pocket Wordstar

Wordstar is the world's most popular Word Processing software. Pocket Wordstar DeLuxe and Pocket Wordstar are versions especially tailored for your 6128 or PCW, and retain all the features that made Wordstar the industry standard. Includes spelling checker and mail merge.



DR Draw

DR Draw lets you create high quality visual aids for business presentations and reports. Using an interactive picture editor, you can draw a wide variety of charts, diagrams and text slides - quickly and easily.



Turn to centre pages to order.

Continued from page 30

You are also prompted for your name and, since the program scans your answer, it is possible to play the role of either a male or female detective. The attitude and answers given by the other characters in the game depend in some measure on your choice.

Despite being classed at an introductory level there is plenty for the seasoned adventurer. Some of the clues tend to be a bit obvious but nothing is quite as straightforward as it seems.

For the beginner it must be the best introduction to the genre I have seen for some time. There are plenty of clues to help you of the "The butler looks meaningfully at the mirror" type and as with all Infocom adventures, it's so well written that it encourages the novice.

The packaging is up to the usual high standard and includes the history of the first castle ghost, the letters, a map and tourist guide to the castle and a Moonmist iron–on T shirt transfer.

Overall, another classic game from the best adventure software house in the world. A bit too easy for the seasoned player but a superb game for the beginner.

Presentation 100%

The best in the business – why don't more companies take a leaf out of their books?

Atmosphere 100%

Positively dripping!

Frustration factor 100%

It's aimed at the beginner and marked accordingly.

Value for money 100%

I wish it was cheaper so that more people would try it. But still worth every penny.

Overall 100%

Can infocom do no wrong?

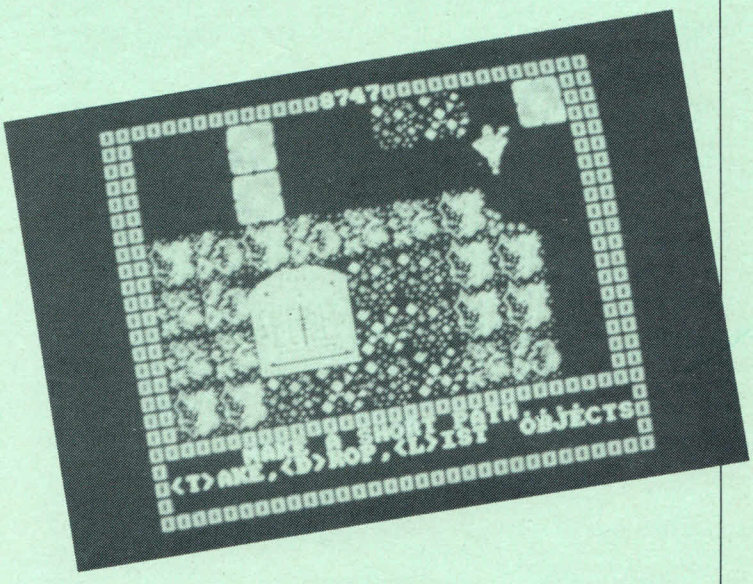
JACKLE & WIDE

Mastertronic; cass, joystick or keys

This is an arcade–style adventure based loosely on the exploits of the characters in Robert Louis Stevenson's classic horror story.

If Dr Jackle had known what he knows now he wouldn't have taken the potion which turned him into the evil Mr Wide. He wouldn't have had to explore the park and the underground sewers in order to find Dr Piqued's antidote.

He wouldn't have had to encounter the monstrous green bottle (Arg! No. Not the green bottle), the intelligent spike or the room of time increase gas (Stevenson was never like this).



Neither would he have had to move his way through Hyde Park on a penny farthing looking for clues (Take object, Drop object, List object) and then explore the underground tunnels and hack his way through to the quarry and the dark forest (aren't they all).

There are a few clues, for example, "a small sea fish is here". My, my, I wonder what that could mean. You can save the game, though, so you do have a chance to find out although it takes ages to save.

It's easy enough to explore the limits of the park but you may need to map the tunnels. The first few were pretty boring – well, I didn't find anything of interest – and you also need to work out what you can do with the objects. I had rather hoped the key would open the lock but perhaps that was the fish, it keeps mentioning.

What I Want to know is: if Wide is so evil why does he want to go back to being Dr Jackle?

If you like graphic arcade adventures and you like trying to piece together clues devised by fiendish programmers then this is a reasonable few dollars' worth. Try it first by all means but don't go out of your way.

Ian Waugh

Presentation 65%
No definable keys. Suck it and see instructions but that's the way with puzzles, isn't it?
Graphics 75%
Nice detail.
Sound 64%
Interesting signature tune plus sound of footsteps.
Playability 55%
Too much trial and error for me.
Addictive qualities 55%
If you get stuck for too long you'll probably just give up.
Value for money 76%
Okay for the money.
Overall 55%
Very much a matter of taste.

BARBARIAN

Palace Software, cassette, joystick or keys

Fight your way past the hype of Page Three Maria Whittaker who may appear on the packaging depending on where you buy this Palace combat game. Ignoring the muscular chappie on the cover who sports one of the wigs rejected by Elton John, and what are you left with? Well, one of the best armed–combat games released for the Amstrad.

Palace have produced a bloody and vicious sword–fighting scenario about which it is difficult to say too much.

Set against four beautifully drawn colour backgrounds, each swordsman has up to 32 different moves. They allow him every possible combination of neck chop, head butt, kick, and chops to neck, body and leg.

While other combat games have shied away from showing the realistic results of metal slicing into unprotected human flesh, Barbarian does quite the opposite.

Heads are severed, blood gushes out and the victor boasts about his conquest to the world.

While I found it great fun to play and an efficient way of ridding yourself of any pent–up aggression. I am in two minds as to whether such a game is basically a good idea, especially when aimed at the younger player.

It could be argued that a nation of young people brought up on video nasties will not find anything distasteful here, but this kind of game does pander to sadistic and aggressive instincts.

There are comical touches, too, as the slimy little green creature drags off the corpses of your slain opponents.

Presentation 90%
Very easy to choose options, one or two players.
Graphics 92%
Superb colour backgrounds, and excellent character animation.
Sound 80%
Nice music, but very simple sound effects.
Playability 90%
Very good joystick response, easy to play.
Addictive qualities 90%
You want to keep playing, but the sight of blood may not appeal to those with good taste.
Value for money 85%
Yet another fighting game, but the most realistic yet.
Overall 90%
Buy it if you enjoy bad taste and have a strong stomach.

Barbarian has all the right programming touches including animated serpents which frame the playing area and switchable music/sound effects.

Separately loaded backgrounds enhance the presentation and there is good joystick control, and superb movements on–screen, apparently modelled from real life action. I wonder, though, how you determine how high blood should spurt from the neck of a severed head.

Victor Laszio

BUBBLER

Ultimate cassette, joystick or keys

Ultimate has always been known for it's 3D adventures, most of which are very much the same. Bubbler recaptures the old style combining arcade action with 3D graphics.

It places you in a geometric world populated by a host of nasty creatures trying to toss you into the abyss which surrounds the area. It also includes a quest.

Dotted around are bubble producers which need to be deactivated by capping them with glass cones. Each screen has several tunnel systems which allow you and your bubble to descend from one level to another, while at the same time adding a cone to your collection.

Presentation 90%

It's nice to see a game with a two–player mode, even if it is only one player at a time.

Graphics 82%

Clear but a little bland.

Sound 25%

Bleep. When will Ultimate realise the Amstrad is not a Spectrum.

Playability 74%

The control method makes the game hard to play.

Addictive qualities 75%

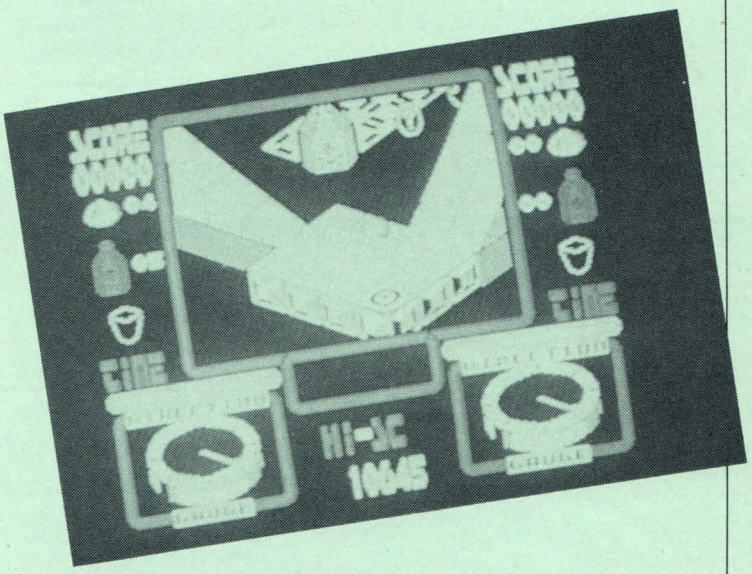
A little frustrating but will have you playing time and again.

Value for money 76%

Reasonable.

Overall 73%

A neat little game for rainy days.



Controlling your bubble is strange. Instead of pushing the joystick in the direction you want to go, as normal, the left and right keys rotate it on the spot and a push forward on the stick moves it in the chosen direction.

The first problems you'll encounter are the jumping spiders which appear from the tops of the bubble producers and pulsating floor pieces. They chase you everywhere but only move slowly from one square to another.

Small towers can fire long–range homing projectiles which you must avoid or forfeit a life. Thankfully once you have been shot by one they blaze away in the wrong direction, and will not become accurate until you begin to move forward.

Not every adversary is deadly – some merely annoy as they try to push you off the edge of the landscape. However, you can defend yourself with high velocity mini bubbles which will destroy all moving objects.

This is Ultimate's best release for over six months and represents the way 3D–type games are going. A lot more could have been done, but it is still a good buy for those of you who enjoy games of skill.

Graphically Bubbler is good, with clearly defined sprites and smooth scrolling but unusually does not have the same "feel" of previous Ultimate games. And the sound leaves a lot to be desired.

It's a mixture of a lot of old ideas, and worth shelling out for if nothing else on the shelves appeals to you.

Anthony Clarke

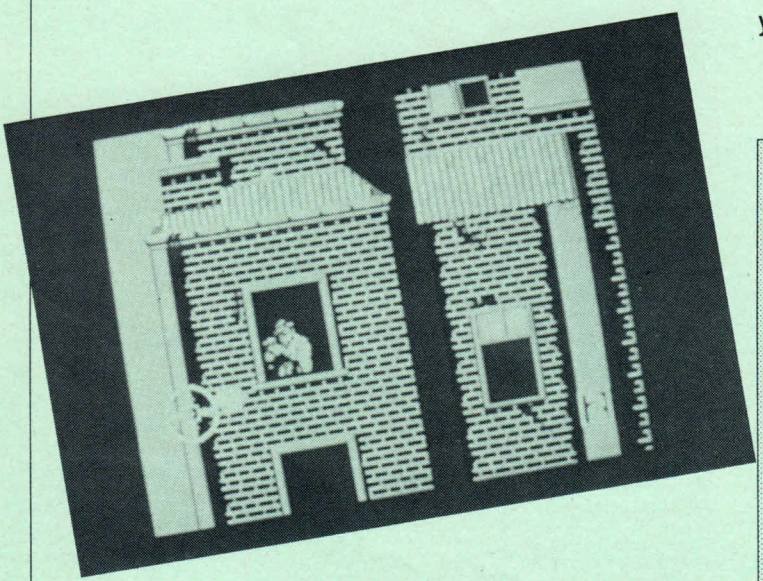
PROHIBITION

Loriciels cass, disc, joystick or keys

After weeks of silence the phone rang. On the other end was Captain Bleak, who sounded troubled. He wanted me to clean up the city.

I crossed town on foot to the dirtiest dive in New York, Deathstone Alley – a place where even gangsters tread softly. I made my way to the top of a building, across the street from Little Hitler's den. Taking out my rifle I prepared to kill.

It's prohibition time in the late 1920's when gangsters ruled. The game is deceptively simple. You must move your sights around a scrolling background, finding and shooting gangsters. The problems become evident when you realise that the entire playing area covers around 20 screens.



Each target gangster has a number of hiding places. Some shoot from windows, behind car doors or the roof. Others can be found standing in the street or leaning on a corner.

While cleaning up you may confront women hostages being used as a shield by the more ruthless gangsters. Random shooting won't work here, instead you must aim carefully at the gangsters head and fire.

Due to the size of the playing area you are given a helping hand in finding the target – an arrow appears and points in the direction of the gangster.

Once all the street gangsters have been despatched you enter the gang's lair.

Success is rewarded by a bonus dependent on how many times you ducked out of the way of the bullets, and a chance to clean up a further area of the alley.

Prohibition comes in two versions, one for the 464 and 664 and an improved version for the 6128.

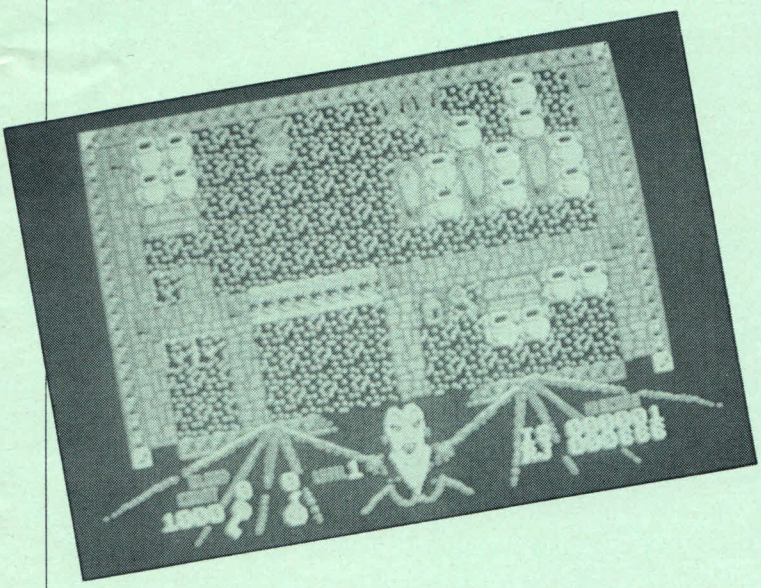
This includes an extended playing area, larger screen display, fancy character set and two pieces of music. These extras may make the 6128 version superior, but the 464 and 664 versions are still great to play.

The graphics are some of the best on Amstrad with colourful Mode backgrounds and well–defined realistic gangster characters.

This isn't quite the best game I've ever played, but it is very well programmed, offers a good blast and will have you coming back time and again.

Anthony Clarke

Presentation 95%
All the trimming of a truly great game.
Graphics 98%
You would be hard pushed to improve them
Sound 81%
Two good tunes and some interesting bullet effects.
Playability 85%
Easy to control, slightly harder to master.
Addictive qualities 79%
You"ll be back for more.
Value for money 75%
A little pricey but worth the extra for a good game.
Overall 86%
One of the best "spirit lifting" games since Ghostbusters.



MR WEEMS AND THE SHE-VAMPIRES

Piranha,cass, disc, joystick or keys

Having failed as an accountant and finding life as a lion tamer palling rather than appalling. Weems turns his talents to vampire hunting. In the same vein as Gauntlet your task is to guide him through six levels of the Great She–Vampire's mansion.

The object is to collect a weapon on each level – crucifix, bible, mallet, stake and so on. All are needed if you are to succeed in the final conflict with the Great She–Vampire.

On your travels you are constantly assailed by vampire bats, she– vampires and Frankenstein monsters, who fancy a swift pint of the red stuff.

Each level is spread over several screens and encounters with any of the toothies drains your blood count, eventually leading to your death.

Fortunately bottles of blood are scattered around which can top up your fluid level. Also to be picked up are garlic bombs and keys to numerous doors.

The bombs clear a screen of everything unfriendly apart from she–vampires which you must despatch with a few well–placed shots from your trusty garlic gun.

When a screen has been cleared either by bomb or gun you are free to drill the coffins full of garlic, which stops them producing a new wave of nasties when you re-enter the screen.

Matters are further complicated with hidden doors, transporters and on level five taking the wrong exit from the screen dumps you back at the start of the level. I never managed to find my way to level six.

Graphics are executed in Mode 1 for detail but a rather dingy choice of colours dulls the effect and it soon gets tiring on the eyes. Sound is restricted to the fluttering of leathery wings plus a few bleeps, bangs and tinkles.

The colours change on each level but the scenery is very similar and this contributes to a lack of visual appeal. But we all know gameplay is what counts and I'm afraid the game doesn't shine here either.

As a full–price game it's competing with Gauntlet, Rana Rama and the likes of Arkanoid, Head over Heels and Starglider. I'd only opt for Weems if you're an addict of the genre.

Ian Sharpe

Presentation 50%
Definable keys and variable sill level.
Graphics 60%
Detailed but not always clear and lack colour.
Sound 10%
The sound FX man must've been off with a sore throat.
Playability 55%
Challenging but repetitive.
Addictive qualities 50%
Yes, but not for long.
Value for money 30%
A budget game at full price.
Overall 42%

SPINDIZZY

(Electric Dreams)

Instead of using the brake, boost in the opposite direction.

GREEN BERET

(Imagine)

If the man is behind you, scroll the screen jump up and stab him in the back. To deal with karate troops, lie down and stab them in the feet or wait until they jump and then stab them.

BOMBSCARE

(Firebird)

The dynamite is used to destroy all the nasties in the room. To help you get around quickly, here are some codes; ZEPHA, DELTA, and QUART. Always make a map; start in the bomb room.

JACK THE NIPPER

(Gremlin)

Get the horn. Enter the police station. Honk at the cat. Wait until both men are on the right of the screen, then honk at them. If you get chased, leave and re–enter.

GHOSTS 'N' GOBLINS

(Elite)

On the bridge, try ducking. On the bouncing knights, shoot each one as this delays the next.

KNIGHT TYME

(Mastertronic)

Sharon likes chocolate. To order some people around you need a valid ID card. To get one.

- Tell Derby IV to help
- Pick up a blank ID card
- Get camera and film to S3 E3 or Klink
- Tell him to help
- Take photo
- Get glue from S3 E3.

You now have a valid card. Derby IV has a chocolate heart. To get things from the top of doors, drop the advert and use it to help you up.

Easydraw

Your electronic brush & canvas

Easydraw is a powerful graphics utility which you can use to create your own spectacular titles, pictures or games backgrounds. You can even build up a picture, save it part-finished to tape or disc, reload it at a later date and improve it until it suits your needs.

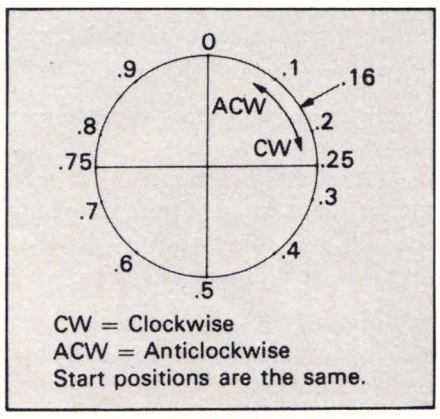


Figure 1

The picture you have created and saved can be used in your own programs either by loading straight to the screen or loading into the reserved memory for recall when needed.

The off–screen pictures shown on these pages took only minutes to produce, but of course it will take you longer until you become familiar with the facilities available.

A brief resume of the capabilities of each option is shown in Table I.

You would be well advised to read the instructions in stages. There are 19 options and each needs practice to understand it fully and become proficient with its use. Read the guide to Easy Draw's functions down to the Beam option first. Then try drawing different coloured horizontal, vertical and diagonal lines. Once you understand these first five functions, build upon your knowledge by reading each subsequent option and practising it until you understand them all.

The following tips on drawing circles and polygons will help to make life a little easier for the first time user:

• Estimating the start position for plotting circles and polygons can present some problems. In the early stages these calculations can be made much easier with the use of a visual aid.

Draw a 100mm diameter circle on a piece of clear plastic using a compass and a felt tip pen. Then mark the circle as a Figure I. When estimating the start position for a plot, hold the plastic with the O and .5 vertical near the screen and read off the position required.

Some interesting shapes can be obtained quite simply by using the part shape and directional plot options as shown in Figures II and III.

• You can change the colour while plotting to produce a multicoloured shape. Choose a high number of sides to give you more time (there is

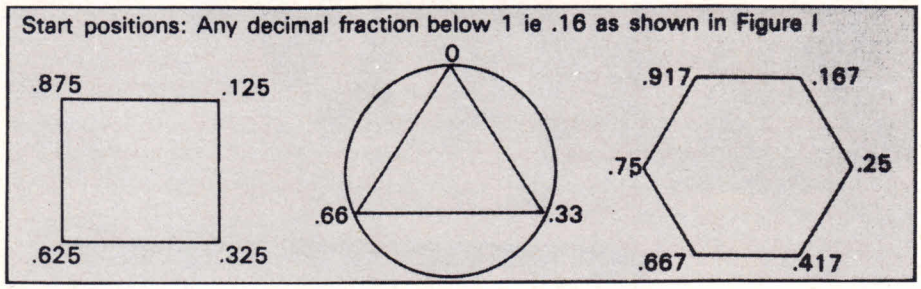


Figure II

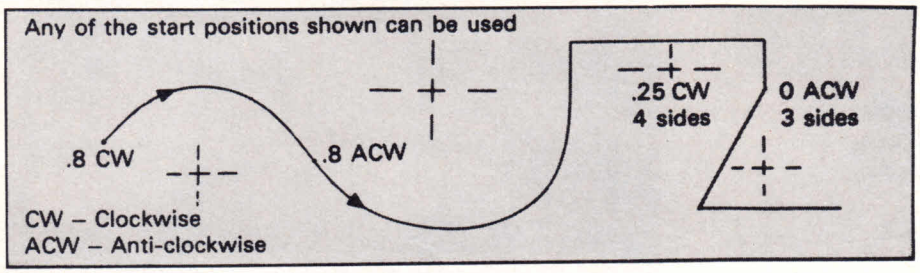


Figure III

no limit). A polygon with 1,000 sides plots slowly and gives you time to change colours or stop the plot wherever you wish.

• Draw to Tab can be implemented when drawing circles and will produce a cone shape, whereas setting Tab Move before entering the circle option and then Drawing to Tab will produce cylindrical shapes.

• If you make a mistake when drawing a shape you can erase it immediately by selecting the background colour and repeating the process you have just used to create the shape. However it can be difficult to remember exactly the sizes and positions used. A much easier way is to set Tab at one side of the shape, turn on the Draw to Tab function and as the cursor is moved the shape will be erased.

• To save a shape to memory you can draw directly to the screen freehand. Alternatively you can prepare a grid (Draw to Tab + Tab Move) to guide you. Change the drawing colour, set Tab, move to the next tab setting and draw a line to the previous tab then set the next Tab. In this way the shape is "memorised" while it is being drawn. Draw it at about half the size of the screen for the best results. A large shape reduces far better than a small one enlarges.

• To exit from Easy Draw press Ctrl+? as if you were going to save the screen to memory, but answer N to the first prompt and the option to End will be given. Escape can only be used in the main program when an input is required – such as Quick Circle – so this is the ideal way to access your listing to search for typing errors.

To run the program without the opening screen use RUN 200. If the program stops as a result of a typing error and you are halfway through your first masterpiece don't panic! Put on the kettle, enter GOTO 550 the main program, and save the picture before you do anything else.

• Use a light touch in the Memory option as a prolonged keypress toggles the memory on and off quickly and will result in empty shapes being numbered in memory.

• Before saving any picture remove the flashing cursor by either moving it off the screen or by drawing a line in the colour of that position.

• When changing INKs make a note of the command given at the time as these are needed to be set up in our own program. Avoid changing INK 15 in (P): INK 4 in (E) and INK 1 in (B) as these are used for screen messages and you may end up with one you can't see. Change them once your drawing is finished to any INK you wish.

• Several options use the same line for messages, so remember to switch them off when you have finished with them if you are using two or three at once.

• You can incorporate pictures created with Easydraw in your own programs. When a picture is saved the whole screen, not just the window, is saved. The border around the window is, however, saved in the background colour. So if you recall a screen in one of your games the border will be available for on–screen prompts.

A saved picture can be loaded directly to screen when, of course, it will be displayed immediately. Listing II shows the idea.

Alternatively you can load the picture into memory that's not screen memory. When you want to display it a simple machine code routine will move it from storage into screen memory.

This has the advantage that you can move your stored picture onto the screen, manipulate it, and resave it, as in Listing III.

User's Guide to Easy Draw's functions

Colour choice

To change the drawing colour press Shift + the corresponding letter key opposite the colour block.

Ink change

Press Ctrl + P followed by the appropriate letter key and Enter. The selected colour will turn black, at which point pressing Shift will move through the colour palette and Enter will fix the chosen colour.

Cursor control

The arrow keys control the cursor, and coupled with Shift will jump the cursor 20 pixels for faster movement.

Cursor jump

Ctrl + J will alter the size. Enter the horizontal and vertical size.

Beam

Ctrl + B toggles Beam on/off to draw lines, and cursor jump can be implemented. Alternatively Ctrl + Copy toggles the Copy key to draw but only when it is held down. With the latter option cursor jump cannot be implemented to draw – only to move.

Tab setting

The tab key "sets" the cursor position which is memorised and indicated in yellow above the permanently displayed X, Y coordinates of the current cursor.

Lines can be drawn to Tab from any position, either on or off the screen.

Draw single line to Tab

Shift + Tab draws a line from the current cursor position to the previously set Tab position.

Draw to Tab

Ctrl + D toggles this option to draw a line to Tab with every move of the cursor – including any cursor jump implementation. This is useful for drawing several lines to Tab and avoids having to press Shift + Tab for each line as in draw single line.

Fill

Place the cursor inside any shape and Ctrl + F will fill it to the nearest vertical line under arrow key control as the cursor is moved vertically.

Erase

Selecting background colour A with the Beam Draw to Tab, or Fill options will remove foreground colours. For full screen erasure press Shift + Clr.

Quick circle

Use the arrow keys to place the centre, press Ctrl + Q and either enter a radius > 4 or press 0. The latter enables the left or right arrow keys to move a second cursor with which the radius can be set by pressing Enter.

Circles, ellipses and polygons

Use the arrow keys to position the shape, then press Ctrl and Clr and input both horizontal and vertical radii using the same options as a quick circle.

Input the start position, a number between 0 and 1 (.5 will start to draw from the bottom of the shape). This is calculated using decimal fractions on a clockwise circle.

Input the number of sides. If this number is less than 20 you will be prompted for a delay. This is only needed if a part shape is required so that pressing the spacebar can halt the plotting.

Answer 0 or 1 to the prompt for clockwise or anti–clockwise plotting.

Tab move

The Tab setting can be made to move with the cursor by pressing Ctrl + Tab. Coupled with Draw to Tab parallel lines can be drawn using cursor jump to create a grid. This also works in the circle (polygon routine but must be set before Ctrl + Clr.

Shape memory

Ctrl + M stores and numbers up to 40 shapes to be recalled and drawn at any size. Press Tab, and while drawing a continuous line press Tab again at every change in direction (only Tab settings are memorised). Check your shapes using the redraw option before you save using Ctrl + S.

Ctrl + L will load a new shape file from memory.

Redraw shape

Ctrl + R and enter shape number and magnitude required (from .1 upwards), to enlarge or reduce the original.

Type at cursor

Press Ctrl + T to enter text, numbers or Ctrl graphics at the cursor position.

Go back to options

Ctrl + G will return you to the opening options for a mode change or to load a picture from memory.

Save screen

Ctrl + ? saves the screen to tape or disc.

Load screen

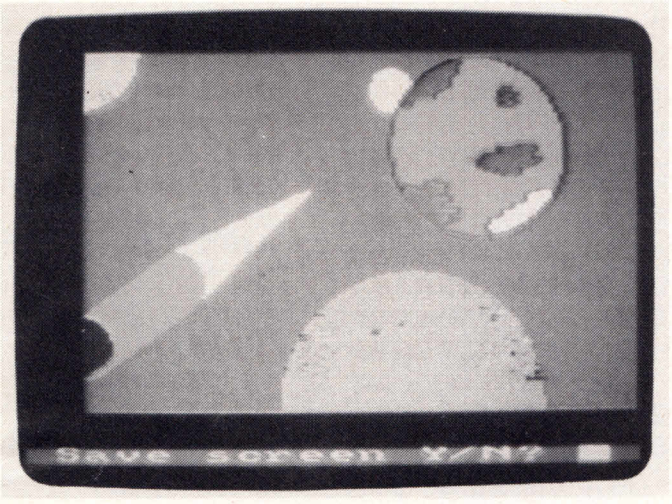
A saved screen can be loaded into memory using either the option at the start of the program, or Ctrl + L at any time.

```
50 REM ************************ EA
SY DRAW ************************
*
60 REM *** A screen drawing and pa
inting utility by Glynne Davies **
*
70 REM
80 REM
90 REM
100 REM
110 MEMORY &665A : REM * reserve m
emory for screen dump and disc dri
ve *
120 GOSUB 3700: REM * load machine
 code for memory dump *
130 REM *** Title page ***
140 MODE 0:FOR n=-PI TO 3 STEP PI/
16:pt= (pt+1) MOD 7:pc=pt+1
150 TAG: PLOT 80+(20*SIN(n)),200+(
120*COS(n)),pc
160 PRINT "*--EASY DRAW--*";:NEXT:
TAGOFF
170 LOCATE 10,23:PRINT "BY":LOCATE
 6,25: PEN 3:PRINT "G.M.DAVIES"
180 FOR n=1 TO 6000:NEXT
190 CALL &BC02
200 MODE 2:INK 15,1:INK 0,13:INK 1
,0:x=286:y=150:beam =0:testab=3
210 jump=20:jumpy=20:DIM blank$(10
0):DIM character$(40)
220 beam=0:LOCATE 17,12:INPUT "Do
you want to load a picture to memo
ry Y or N";pic$
230 IF pic$="Y" OR pic$="y" THEN L
OCATE 20,14:INPUT "Type in the nam
e of the picture";picture$:ELSE GO
TO 250
240 LOAD picture$,&6675
250 CLS:LOCATE 20,12:INPUT "Please
 type in the mode required (ie. 0
 or 1 )";modescreen:IF modescreen
> 1 OR modescreen <0 THEN GOTO
 250
260 IF modescreen =1 THEN movx= 2:
ELSE movx=4
270 IF modescreen >1 THEN GOTO 250

280 MODE modescreen:INK 1,24
290 LOCATE(4*(modescreen*2)+5),12:
 INPUT "PICTURE Y/N";Y$
300 CLS
310 IF Y$="Y" OR Y$="y" THEN CALL
26215
320 GOSUB 360 : REM * initialize *
330 GOSUB 510 : REM * program *
340 END
350 REM *** Initialize - set up th
e screen ***
360 WINDOW #2,1,2,1,22:WINDOW #3,1
,80/movx,23,25:WINDOW #0,1,80/movx
,23,25:WINDOW #4,3,80/movx,1,22
370 PAPER #2,0:PAPER #3,15:CLS #2:
CLS #3
380 ORIGIN 64,48,64,640,48,400
390 DRAW 574,1,3:DRAW 574,350,3:DR
AW 1,350,3:DRAW 1,1,3
400 PEN #2,0:LOCATE #2,2,20:PRINT
#2,CHR$(143);
410 REM ** Draw colours **
420 FOR count=0 TO 15
430 PEN #2,count
440 LOCATE #2,1,count+1:PRINT #2,C
HR$(65+count);CHR$(233)
450 NEXT count
460 PEN #2,4:LOCATE #2,1,1:PRINT #
2,"A"
470 PEN #2,5:LOCATE #2,1,17:PRINT
#2,"+"
480 IF movx=2 THEN LOCATE #2,1,1:P
RINT #2,"A":LOCATE #2,1,5:PRINT #2
,"E":LOCATE #2,1,9:PRINT #2,"I":LO
CATE #2,1,13:PRINT #2,"M"
490 shift$="SHIFT":FOR count=1 TO
5:LOCATE #2,1,17+count:PRINT #2,MI
D$ (shift$,count,1):NEXT
495 PEN #3,1:LOCATE #3,13,3:PRINT
#3,"Beam off";:REM beam off
500 REM *** program ***
505 ON ERROR GOTO 505
510 WHILE exit <1
520 IF INKEY (1)=0 THEN PLOT x,y,t
es:x=x+movx:IF movtab=1 THEN extra
x=extrax+movx
530 IF INKEY (8)=0 THEN PLOT x,y,t
es:x=x-movx:IF movtab=1 THEN extra
x=extrax-movx
540 IF INKEY (0)=0 THEN PLOT x,y,t
es:y=y+2:GOSUB 2250:IF movtab=1 TH
EN extray=extray+2:REM gosub to fi
ll routine
550 IF INKEY (2)=0 THEN PLOT x,y,t
es:y=y-2:GOSUB 2250:IF movtab=1 TH
EN extray=extray-2
560 IF INKEY (1)=32 THEN PLOT x,y,
tes:x=x+jump:IF movtab=1 THEN extr
ax=extrax+jump
570 IF INKEY (8)=32 THEN PLOT x,y,
tes:x=x-jump:IF movtab=1 THEN extr
ax=extrax-jump
580 IF INKEY (0)=32 THEN PLOT x,y,
tes:y=y+jumpy:IF movtab=1 THEN ext
ray=extray+jumpy
590 IF INKEY (2)=32 THEN PLOT x,y,
tes:y=y-jumpy:IF movtab=1 THEN ext
ray=extray-jumpy
```

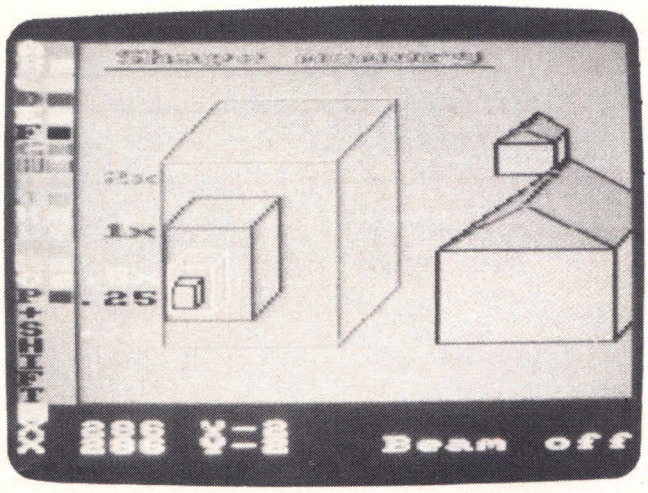
```
600 IF mem=1 AND xp=XPOS AND yp=YP
OS AND memtrip=1 THEN character$(s
h)=character$(sh)+STR$(xrel)+STR$(
yrel):memtrip=0:IF LEN(charact
er$(sh)) >240 THEN SOUND 1,300,25:
 PEN #3,1:LOCATE #3,1,1:PRINT #3,"
Last tab setting ":mem=0
610 IF INKEY(50)=128 THEN SOUND 1,
100,20:GOSUB 2920:REM redraw memor
y shape
620 IF INKEY(36)=128 THEN SOUND 1,
100,15:GOSUB 3520: REM load pictur
e into memory
630 IF INKEY(16)=32 THEN CLS #4:MO
VE 0,0:DRAW 574,1,3:DRAW 574,350,3
:DRAW 1,350,3:DRAW 1,1,3:MOVE x,y:
REM * erase picture *
640 IF INKEY (9)=128 THEN SOUND 1,
120,10:beamtrip=(beamtrip+1) MOD 2
:IF beamtrip=0 THEN PEN #3,1:LOCAT
E #3,13,3:PRINT #3,"Beam off";
:REM beam on/off/on copy key
650 IF beamtrip=0 THEN GOTO 670
660 IF INKEY (9)=0 THEN beam=1:PEN
 #3,4:LOCATE #3,13,3:PRINT #3,"Bea
m on ";:ELSE beam=0:PEN #3,4:LOCAT
E #3,13,3:PRINT #3,"Beam off";
:REM beam on/off/on copy key
670 IF beamtrip=1 THEN GOTO 690
680 IF INKEY (54)=128 THEN SOUND 1
,100,10:beam=(beam+1) MOD 2:IF bea
m=1 THEN PEN #3,1:LOCATE #3,13,3:P
RINT #3,"Beam on ";:ELSE beam=
0:PEN #3,1:LOCATE #3,13,3:PRINT #3
,"Beam off";:REM beam on/off
690 IF INKEY(68)=128 THEN SOUND 1,
100,20:movtab=(movtab+1) MOD 2:IF
movtab=1 THEN PEN #3,1:LOCATE #3,1
,2:PRINT #3,"Tab moving      "
:ELSE PEN #3,1: LOCATE #3,1,2:PRIN
T #3,"Tab stopped     ":xp=xp+extr
ax:yp=yp+extray:extrax=0:extray=0
:REM moving tab
700 IF INKEY(35)=128 THEN SOUND 1,
100,10:GOSUB 3350 : REM input from
 tape
710 IF INKEY (21)=32 THEN GOSUB 10
40: REM * pen colours *
730 IF INKEY (45)=128 THEN SOUND 1
,100,25:GOSUB 2800: REM cursor jum
p
740 IF INKEY (68)=0 THEN SOUND 1,1
00,5:xrel=x-xp:yrel=y-yp:xp=x:yp=y
:PEN #3,1:LOCATE #3,1,2:PRINT #3,"
X";XPOS;"Y";YPOS:tb=1:memtrip=
1:REM tab setting
750 IF sh> 39 THEN GOTO 770
760 IF INKEY (38)=128 THEN SOUND 1
,100,5:mem=(mem+1) MOD 2:IF mem=1
THEN PEN #3,1:LOCATE #3,1,1:PRINT
#3,"Memory on shape";sh:ELSE P
EN #3,1:LOCATE #3,1,1:PRINT #3,"Me
mory is off      ":sh=sh+1
770 IF INKEY (61)=128 THEN SOUND 1
,100,5:fil=(fil+1) MOD 2:IF fil=1
THEN LOCATE #3,1,1:PRINT #3,"Draw
to tab on  ":ELSE LOCATE #3,1,
1:PRINT #3,"Draw to tab off ": REM
 draw to tab function
780 IF INKEY(52)=128 THEN SOUND 1,
200,25:GOSUB 3630:REM go to option
s pages
790 IF INKEY (60)=128 THEN SOUND 1
,100,5: GOSUB 3230 : REM save shap
es to tape
800 IF INKEY (67)=128 THEN SOUND 1
,100,25:PLOT x,y,14:GOSUB 960 :REM
 * Quick circle *
810 IF INKEY (68)=32 THEN DRAWR xp
+extrax-XPOS,yp+extray-YPOS,p :PLO
T x,y,14
820 IF INKEY (16)=128 THEN SOUND 1
,100,10:PLOT x,y,14: GOSUB 1230 :
REM * circles, polygons *
830 IF INKEY (53)=128 THEN SOUND 1
,100,5:infil=(infil+1) MOD 2:IF in
fil=1 THEN filc=TEST(x-2,y-2):LOCA
TE #3,1,1:PRINT #3,"Fill on
      ":ELSE LOCATE #3,1,1:PRINT #
3,"Fill off      "
840 IF INKEY(27)=128 THEN SOUND 1,
100,5: GOSUB 2450 :REM * ink colou
r change *
850 IF INKEY (30)=128 THEN SOUND 1
,100,5:MOVE 0,0:DRAW 574,1,3:DRAW
574,350,3:DRAW 1,350,3:DRAW 1,1,3:
GOTO 2650 : REM * picture save
 *
860 IF INKEY(51)=128 THEN SOUND 1,
100,5:GOSUB 2070 : REM * type at c
ursor *
870 IF x=xx AND y=yy THEN GOTO 940
880 IF beam=1 THEN DRAW x,y,p:tes=
p
890 IF beam=0 THEN tes=TEST(x,y):x
x=x:yy=y:PLOT x,y,14
900 IF tb=1 AND fil =1 THEN DRAWR
 xp+extrax-XPOS,yp+extray-YPOS,p
910 PEN #3,4:LOCATE #3,1,3:PRINT #
3,"X";x;"Y";y
920 IF xp=xpt AND yp=ypt AND extra
x=xextra AND extray=yextra THEN GO
TO 940
930 PEN #3,1:LOCATE #3,1,2:PRINT #
3,"X";xp+extrax;"Y";yp+extray :xpt
=xp:ypt=yp:xextra=extrax:yextra=ex
tray
```

```
940 WEND
950 REM ** Quick circle **
960 CALL &BB1B:IF INKEY(67)=128 TH
EN GOTO 960:CALL &BB1B
970 qc=1:PEN #3,1:LOCATE #3,1,1:IN
PUT #3,"RADIUS<-> above 4 or curso
r enter 0";radx$:IF radx$=" " OR r
adx$="" THEN GOTO 970
975 IF ASC(radx$) < 48 THEN GOTO 9
70
980 radx= VAL(radx$):LOCATE #3,1,1
:PRINT #3,SPACE$ (40):IF radx=0 TH
EN GOSUB 1850
990 radx=radx+(radx MOD movx) :MOV
E x+radx,y
1000 FOR count=0 TO 360 STEP 2:DEG
:DRAW x+(radx*COS(count)),y+(radx*
SIN(count)),p:NEXT
1010 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
1020 RAD:qc=0:RETURN
1030 REM ** pen colour change **
1040 IF INKEY (69)=32 THEN SOUND 1
,150,5:p=0 :REM * A *
1050 IF INKEY (54)=32 THEN SOUND 1
,150,5:p=1 :REM * B *
1060 IF INKEY (62)=32 THEN SOUND 1
,150,5:p=2 :REM * C *
1070 IF INKEY (61)=32 THEN SOUND 1
,150,5:p=3 :REM * D *
1080 IF INKEY (58)=32 THEN SOUND 1
,150,5:p=4 :REM * E *
1090 IF INKEY (53)=32 THEN SOUND 1
,150,5:p=5 :REM * F *
1100 IF INKEY (52)=32 THEN SOUND 1
,150,5:p=6 :REM * G *
1110 IF INKEY (44)=32 THEN SOUND 1
,150,5:p=7 :REM * H *
1120 IF INKEY (35)=32 THEN SOUND 1
,150,5:p=8 :REM * I *
1130 IF INKEY (45)=32 THEN SOUND 1
,150,5:p=9 :REM * J *
1140 IF INKEY (37)=32 THEN SOUND 1
,150,5:p=10:REM * K *
1150 IF INKEY (36)=32 THEN SOUND 1
,150,5:p=11:REM * L *
1160 IF INKEY (38)=32 THEN SOUND 1
,150,5:p=12:REM * M *
1170 IF INKEY (46)=32 THEN SOUND 1
,150,5:p=13:REM * N *
1180 IF INKEY (34)=32 THEN SOUND 1
,150,5:p=14:REM * O *
1190 IF INKEY (27)=32 THEN SOUND 1
,150,5:p=15:REM * P *
1200 PEN #2,p:LOCATE #2,2,20:PRINT
 #2,CHR$(233);
1210 RETURN
1220 REM *** circles ***
1230 CALL &BB1B:IF INKEY(16)=128 T
HEN GOTO 1230:CALL &BB1B
1240 PEN #3,10:LOCATE #3,1,1:INPUT
 #3,"RADIUS<-> above 4 or cursor e
nter 0";radx$:IF radx$= " " OR rad
x$="" THEN GOTO 1240
1245 IF ASC(radx$) < 48 THEN GOTO
1240
1250 radx= VAL(radx$)
1260 LOCATE #3,1,1:PRINT #3,SPACE$
 (40):IF VAL(radx$) < 4 THEN GOSUB
 1850:GOTO 1300
1270 LOCATE #3,1,1:PRINT #3,SPACE$
 (40)
1280 LOCATE #3,1,1:INPUT #3,"RADIU
S up/down >2";rady$:IF rady$=" " O
R rady$="" THEN GOTO 1280
1285 IF ASC(rady$) < 48 THEN GOTO
1280
1290 rady=VAL(rady$):IF rady <2 TH
EN GOSUB 1940
1300 LOCATE #3,1,1:PRINT #3,SPACE$
 (60)
1310 LOCATE #3,1,1:INPUT #3,"Start
 position 0 to 1 (ie .65)";stpos$:
IF stpos$=" " OR stpos$="" THEN GO
TO 1310
1315 IF ASC(stpos$) < 46 THEN GOTO
 1310
1320 stpos=VAL(stpos$):IF VAL(stpo
s$) >1 THEN LOCATE #3,1,1:PRINT #3
,"BETWEEN 0 and 1 ie. .1 .25 .7 .9
":FOR n=1 TO 1000:NEXT:GOTO 13
10
1330 LOCATE #3,1,1:INPUT #3,"Numbe
r of sides. Go >70 for circles";si
de$:IF side$=" " OR side$="" THEN
GOTO 1330
1335 IF ASC(side$) <48 THEN GOTO 1
330
1340 side=VAL(side$):IF VAL(side$)
 < 3 THEN GOTO 1330
1350 poly=(2*PI)/side
1360 LOCATE #3,1,1:PRINT #3,SPACE$
 (40)
1365 IF side > 19 THEN GOTO 1380 :
REM miss delay
1370 IF side < 20 THEN LOCATE #3,1
,1:INPUT #3,"state delay ie. 200 f
or part shape";delay$:IF delay$="
" OR delay$="" THEN GOTO 1370
1375 IF side < 20 AND ASC(delay$)
< 48 THEN GOTO 1370: ELSE delay=VA
L(delay$):LOCATE #3,1,1:PRINT #3,
SPACE$(40)
1380 LOCATE #3,1,1:INPUT #3,"Clock
wise plotting 0 Anticlockwise 1";c
w$:IF cw$=" " OR cw$="" THEN GOTO
```



```
1380
1385 IF ASC(cw$) <48 THEN 1380
1390 cw=VAL(cw$):IF VAL(cw$) >1 TH
EN GOTO 1380
1400 LOCATE #3,1,1:PRINT #3,SPACE$
 (40)
1410 LOCATE #3,1,1:INPUT #3,"Plot
from the cursor 1 or the centre 0"
;startat$:IF startat$=" " OR start
at$="" THEN GOTO 1410
1415 IF ASC(startat$) < 48 THEN GO
TO 1410
1420 startat=VAL(startat$):IF VAL(
startat$) >1 THEN GOTO 1410
1430 IF tb=0 THEN GOTO 1460
1440 LOCATE #3,1,1:INPUT #3,"Draw
to tab setting 1 or 0 for off (3D)
";tba$:IF tba$=" " OR tba$="" THEN
 GOTO 1440
1445 IF ASC(tba$) <48 THEN GOTO 14
40
1450 tba=VAL(tba$):IF VAL(tba$) >
1 THEN GOTO 1440
1460 LOCATE #3,1,1:PRINT #3,SPACE$
 (40)
1470 LOCATE #3,1,1:PRINT #3,"SPACE
 BAR TO STOP"
1475 radx=radx+(radx MOD movx)
1480 IF cw=1 THEN GOTO 1680: REM a
nticlockwise
1490 IF startat=0 THEN PLOT x,y,te
s
1500 IF startat=1 THEN x=x-(radx*
SIN(stpos*2*PI)):y=y-(rady*COS(stp
os*2*PI)):MOVE x,y
1510 MOVE x+(radx*SIN(stpos*2*PI))
,y+(rady*COS(stpos*2*PI))
1520 REM ** circle clockwise **
1530 FOR count= stpos*2*PI TO (4*P
I)+poly STEP poly
1540 DRAW x+(radx*SIN(count)),y+(r
ady*COS(count)),p:IF movtab=1 THEN
 extrax=(XPOS-x)-radx*SIN(stpos*2*
PI):extray=(YPOS-y)-rady*COS(s
tpos*2*PI)
1550 IF INKEY (68)=32 THEN DRAWR x
p+extrax-XPOS,yp+extray-YPOS,p:MOV
E x+(radx*SIN(count)),y+(rady*COS(
count))
1560 IF INKEY (21)=32 THEN GOSUB 1
040: REM * pen colours *
1570 IF tba=1 THEN DRAWR xp+extrax
-XPOS,yp+extray-YPOS,p:MOVE x+(rad
x*SIN(count)),y+(rady*COS(count))
1580 IF INKEY (47)=0 THEN count=(4
*PI)+poly
1590 IF xyoff=1 THEN GOTO 1610
1600 LOCATE #3,1,3:PRINT #3,"X";XP
OS;"Y";YPOS
1610 FOR rest=0 TO delay:IF INKEY
(47)=0 THEN count=(4*PI)+poly:ELSE
 NEXT
1620 NEXT
1630 LOCATE #3,1,1:PRINT #3,SPACE$
(60):x=XPOS:y=YPOS
1640 LOCATE #3,1,3:PEN #3,4:PRINT
#3,"X";XPOS;"Y";YPOS:IF beam=1 THE
N PEN #3,1:LOCATE #3,13,3:PRINT #3
,"Beam on ";:ELSE beam=0:PEN #
3,1:LOCATE #3,13,3:PRINT #3,"Beam
off";:REM beam on/off
1650 extrax=CINT(extrax):extray=CI
NT(extray)
1660 RETURN
1670 REM ** circle anticlockwise *
*
1680 stpos=stpos+0.5:IF startat=0
THEN PLOT x,y,tes
1690 IF startat=1 THEN x=x+(radx*
SIN(stpos*2*PI)):y=y+(rady*COS(stp
os*2*PI))
1700 MOVE x-(radx*SIN(stpos*2*PI))
,y-(rady*COS(stpos*2*PI))
1710 FOR count=2*PI*stpos TO -((2*
PI)+poly) STEP -poly
1720 DRAW x-(radx*SIN(count)),y-(r
ady*COS(count)),p :IF movtab=1 THE
N extrax=(XPOS-x)+radx*SIN(2*PI*st
pos):extray=(YPOS-y)+rady*COS(
2*PI*stpos)
1730 IF INKEY (68)=32 THEN DRAWR x
p+extrax-XPOS,yp+extray-YPOS,p:MOV
E x-(radx*SIN(count)),y-(rady*COS(
count))
1740 IF tba=1 THEN DRAWR xp+extrax
-XPOS,yp+extray-YPOS,p:MOVE x-(rad
x*SIN(count)),y-(rady*COS(count))
```

```
1750 IF INKEY (47)=0 THEN count= -
((2*PI)+poly)
1760 LOCATE #3,1,3:PRINT #3,"X";XP
OS;"Y";YPOS
1770 IF INKEY (21)=32 THEN GOSUB 1
040: REM * pen colours *
1780 FOR rest=0 TO delay:IF INKEY
(47)=0 THEN count= -((2*PI)+poly):
ELSE NEXT
1790 NEXT
1800 LOCATE #3,1,1:PRINT #3,SPACE$
(60):x=XPOS:y=YPOS:count =0
1810 LOCATE #3,1,3:PEN #3,4:PRINT
#3,"X";XPOS;"Y";YPOS:IF beam=1 THE
N PEN #3,1:LOCATE #3,13,3:PRINT #3
,"Beam on ";:ELSE beam=0:PEN #
3,1:LOCATE #3,13,3:PRINT #3,"Beam
off";:REM beam on/off
1820 extrax=CINT(extrax):extray=CI
NT(extray)
1830 RETURN
1840 REM * horizontal cursor radiu
s setting *
1850 WHILE radx =0
1860 LOCATE #3,1,1:PRINT #3,"Use a
rrow keys <--> to place,then enter
";
1870 IF INKEY(1)=0 THEN PLOT x+rst
ep,y,rtes:rtes=TEST(x+rstep+movx,y
):rstep=rstep+movx:PLOT x+rstep,y,
14
1880 IF INKEY (8)=0 THEN PLOT x+rs
tep,y,rtes:rtes=TEST(x+rstep-movx,
y):rstep=rstep-movx:PLOT x+rstep,y
,14
1890 IF INKEY (18)=0 THEN radx=ABS
(rstep):PLOT x+rstep,y,rtes
1900 PLOT x,y,14
1910 WEND:rtes=14:rstep=0:CALL &BB
18
1920 IF qc=1 THEN RETURN
1930 LOCATE #3,1,3:PRINT #3,SPACE$
(20):LOCATE #3,1,3:PRINT #3,"Hori-
rad is";radx;"y=X"
1940 rady=0
1950 REM * up/down radius set by c
ursor *
1960 WHILE rady=0
1970 LOCATE #3,1,1:PRINT #3,"Use a
rrow keys then press enter "
1980 IF INKEY(0)=0 THEN PLOT x,y+r
step,rtes:rtes=TEST(x,y+rstep+2):r
step=rstep+2:PLOT x,y+rstep,14
1990 IF INKEY(2)=0 THEN PLOT x,y+r
step,rtes:rtes=TEST(x,y+rstep-2):r
step=rstep-2:PLOT x,y+rstep,14
2000 IF rstep=0 AND INKEY(63)=0 TH
EN rady=radx
2010 IF INKEY(18)=0 THEN rady=ABS(
rstep):PLOT x,y+rstep,rtes
2020 PLOT x,y,14
2030 WEND:rstep=0:rtes=0:CALL &BBO
0
2040 CALL &BB18:RETURN
2050 REM *** type at cursor ***
2060 CALL &BB1B
2070 LOCATE #3,1,1:PRINT #3,"   Ty
pe at cursor"
2080 LOCATE #3,1,1:PRINT #3,"   Ty
pe at cursor"
2090 LOCATE #3,1,2:PRINT #3,"   (E
nter) to end"
2100 FOR n=1 TO 500:NEXT:CALL &BBO
0
2110 WHILE t<1
2120 type$=INKEY$
2130 IF INKEY(18)=0 THEN PLOT x,y,
0: t=1:papr=0:GOTO 2200
2140 TAG:PLOT x,y,p
2150 IF type$="" GOTO 2190
2160 IF ASC(type$)=127 THEN PLOT x
,y,0: x=x-(movx*4):MOVE x,y:PRINT
" ";:GOTO 2190
2170 PRINT type$;
2180 x=x+(movx*8):MOVE x,y
2190 type$=""
2200 WEND
2210 t=0:LOCATE #3,1,1:PRINT #3,SP
ACE$(40)
2220 TAGOFF
2230 RETURN
2240 REM ** fill routine **
2250 IF infil = 0 THEN RETURN
2260 WHILE lin < 1
2270 IF TEST (x,y) <> filc THEN li
n =1:GOTO 2330
2280 incl=incl+movx:telf=TEST (x-i
ncl,y)
2290 IF telf <> filc THEN MOVE x,y
:DRAW x-(incl-movx),y,p:lin =1
2300 IF x<2 THEN lin=1
2310 IF y<2 THEN lin =1
2320 IF y>348 THEN lin=1
2330 WEND
2340 lin=0
2350 WHILE lin < 1
2360 incr=incr+movx:terf=TEST (x+i
ncr,y)
2370 IF x>575 THEN lin=1
2380 IF y<2 THEN lin=1
2390 IF y>348 THEN lin=1
2400 IF terf <> filc THEN MOVE x,y
:DRAW x+(incr-movx),y,p:lin =1
2410 WEND
2420 lin=0:incl=0:incr=0:telf=17:t
erf=17
```




```
2430 RETURN
2440 REM ** ink change **
2450 CALL &BB1B:IF INKEY(27)=128 T
HEN 2450:CALL &BB1B
2455 LOCATE #3,1,1:INPUT #3,"Ink c
hange Y/N"; y$:IF y$="Y" OR y$="y"
 THEN GOTO 2460:ELSE LOCATE #3,1,1
:PRINT #3,SPACE$(40):RETURN
2460 LOCATE #3,1,1:INPUT #3,"   Pe
n to change";in$:IF in$=""  THEN G
OTO 2460 2470 IF ASC(in$) < 65 OR
ASC(in$) >112 THEN LOCATE #3,1
,1:PRINT #3,SPACE$(40):GOTO 2450
2470 IF ASC(in$) < 65 OR ASC(in$)
>112 THEN LOCATE #3,1,1:PRINT #3,S
PACE$(40):GOTO 2450
2480 in$=UPPER$(in$):IF ASC(in$) >
 80 THEN GOTO 2450
2490 p=ASC (in$)-65
2500 IF p>15 THEN GOTO 2450
2510 CLS #3
2520 WHILE inchange <1
2530 LOCATE #3,1,1: PRINT #3,"Ink
change (Shift)"
2540 PEN #3,p
2550 LOCATE #3,4,2: PRINT #3,"To s
et (Enter)"
2560 LOCATE #3,1,3: PRINT #3,"
 INK";p;",";i;
2570 IF INKEY(21)=32 THEN i=(i+1)
MOD 27
2580 IF INKEY(18)=0 OR INKEY (6)=0
 THEN inchange=1
2590 INK p,i
2600 WEND
2610 i=0:inchange=0
2620 CLS #3:PEN #3,4:LOCATE #3,1,3
:PRINT #3,"X";XPOS;"Y";YPOS
2630 RETURN
2640 REM ** save screen to memory
**
2650 FOR n=1 TO 100:NEXT:CALL &BBO
0
2660 PAPER #2,15:CLS #2:CLS #3
2670 CALL 26203
2680 CLS #4
2690 FOR n=1 TO 2000:NEXT n
2700 CALL 26215
2710 FOR n=1 TO 2000:NEXT
2720 CALL &BB00
2730 LOCATE 2,23:INPUT"Save screen
 Y/N";ipt$
2740 IF ipt$="Y" OR ipt$="y" THEN
GOSUB 2770
2750 MODE 1:y=y+2:LOCATE 12,12:INP
UT "End EASYDRAW Y/N";y$:IF y$="y"
 OR y$="Y" THEN CALL &BC02:MODE 1:
LOCATE 6,2:PRINT "Easydraw pro
gram has finished":END
2755 MODE 2:GOTO 220
2760 REM ** Save to tape **
2770 SPEED WRITE 1:LOCATE 1,23:INP
UT "Please print name  ";name$
2780 SAVE name$,B,&6675,&4000
2790 RETURN
2800 REM *** cursor jump distance
***
2810 CALL &BB1B:IF INKEY(45)=128 T
HEN GOTO 2810:CALL &BB1B
2820 CALL &BB00
2830 PEN #3,1:LOCATE #3,1,1:INPUT
#3," Type in horizontal cursor jum
p";jump$:IF jump$=" " OR jump$=""
THEN GOTO 2830
2835 IF ASC(jump$) <48 THEN GOTO 2
830
2840 IF VAL(jump$)<4 THEN LOCATE #
3,1,1:PRINT #3,SPACE$(40):GOTO 283
0
2850 jump=VAL(jump$)
2860 LOCATE #3,1,1:INPUT #3,"Type
in the vertical cursor jump ";jump
y$:IF jumpy$=" " OR jumpy$="" THEN
 2860
2865 IF ASC(jumpy$) <48 THEN GOTO
2860
2870 IF VAL(jumpy$)<2 THEN LOCATE
#3,1,1:PRINT #3,SPACE$(40):GOTO 28
60
2880 jumpy= VAL(jumpy$)
2890 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
2900 LOCATE #3,1,3:PRINT #3,SPACE$
(20):LOCATE #3,1,3:PEN #3,4:PRINT
#3,"X";XPOS;"Y";YPOS:IF beam=1 THE
N PEN #3,1:LOCATE #3,13,3:PRIN
T #3,"Beam on ";:ELSE beam=0:PEN #
3,1:LOCATE #3,13,3:PRINT #3,"Beam
off";:REM beam on/off
```

```
2910 RETURN
2920 REM draw from memory
2930 CALL &BB1B:IF INKEY(50)=128 T
HEN 2930:CALL &BB1B
2940 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
2950 IF sh=0 THEN LOCATE #3,1,1:PR
INT #3,"No shapes in memory":FOR n
=1 TO 1000:NEXT:LOCATE #3,1,1:PRIN
T #3, SPACE$(40):RETURN
2960 LOCATE #3,1,1:INPUT #3,"Type
shape number";shape$:IF shape$=" "
 OR shape$="" THEN GOTO 2960
2965 IF ASC(shape$) <48 THEN GOTO
2960
2970 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
2980 IF VAL(shape$) >= sh THEN LOC
ATE #3,1,1:PRINT #3,"Not available
 0 to";sh-1:FOR n=1 TO 1000:NEXT:G
OTO 2940
2990 shape= VAL(shape$)
3000 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3010 LOCATE #3,1,1:INPUT #3,"At ma
gnification";mag$:mag= VAL(mag$)
3020 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3030 IF mag < 0.1 THEN LOCATE #3,1
,1:PRINT #3,"A little to small  ":
FOR n= 1 TO 1000:NEXT:GOTO 3010
3040 c=0
3050 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3060 IF INSTR(character$(shape),"
- ")> 1 THEN GOTO 3080
3070 character$(shape)=character$(
shape)+" - "
3080 count=0
3090 WHILE count< (LEN(character$(
shape))-3)
3100 count=count+1
3110 space=INSTR(count,character$(
shape)," "):minus=INSTR(count,char
acter$(shape),"-")
3120 IF space < minus THEN count=s
pace
3130 IF space > minus THEN count=m
inus
3140 blank$(c)=STR$(count):c=c+1
3150 WEND
3160 REM draw from memory
3170 FOR count=2 TO c-3 STEP 2
3180 drax= VAL(MID$(character$(sha
pe),VAL(blank$(count)),VAL(blank$(
count+1))-VAL(blank$(count))))
3190 dray= VAL(MID$(character$(sha
pe),VAL(blank$(count+1)),VAL(blank
$(count+2))-VAL(blank$(count+1))))
3200 DRAWR mag*drax,mag*dray,p
3210 NEXT count
3220 count=0:RETURN
3230 REM save shapes to tape
3240 CALL &BB1B: IF INKEY(60)=128
THEN GOTO 3240:CALL &BB1B
3250 LOCATE #3,1,1:INPUT #3, "Save
 shapes Y/N";satap$:LOCATE #3,1,1:
PRINT #3,SPACE$(40):IF satap$="Y"
OR satap$="y" THEN GOTO 3260:E
LSE GOTO 3340
3260 LOCATE #3,1,1:INPUT #3, "File
 name";file$
3270 IF LEN(file$) > 8 THEN LOCATE
 #3,1,1:PRINT #3,"Below eight lett
ers":FOR n=1 TO 300:LOCATE #3,1,1:
PRINT #3,SPACE$(40):GOTO 3260
3280 OPENOUT file$
3285 PRINT #9,sh
3290 FOR count=0 TO sh
3300 PRINT #9,character$(count)
3310 NEXT count
3320 CLOSEOUT
3330 CLS #3:PEN #3,4:LOCATE #3,1,3
:PRINT #3,"X";XPOS;"Y";YPOS;
3340 RETURN
3350 REM input file for shapes
3360 CALL &BB1B:IF INKEY(35)=128 T
HEN GOTO 3370:CALL &BB1B
3370 LOCATE #3,1,1:INPUT #3, "Load
 shapes Y/N";lotap$:LOCATE #3,1,1:
PRINT #3,SPACE$(40):IF lotap$="Y"
OR lotap$="y" THEN GOTO 3380:E
LSE GOTO 3510
3380 LOCATE #3,1,1:INPUT #3, "File
 name";file$
3390 IF LEN(file$) > 8 THEN LOCATE
 #3,1,1:PRINT #3,"Below eight lett
ers":FOR n=1 TO 300:LOCATE #3,1,1:
PRINT #3,SPACE$(40):GOTO 3380
3400 OPENIN file$
3405 INPUT #9,sh
3410 FOR count =0 TO sh
3420 INPUT #9,character$(count)
3430 NEXT count
3470 FOR count=0 TO sh
3480 IF LEFT$(character$(count),1)
="-" THEN GOTO 3490:ELSE character
$(count)=" "+character$(count)
3490 NEXT count:count=0
3500 CLS #3:PEN #3,4:LOCATE #3,1,3
:PRINT #3,"X";XPOS;"Y";YPOS;
3510 RETURN
3520 REM load picture to memory
3530 CALL &BB1B:IF INKEY(36)=128 T
HEN GOTO 3530: CALL &BB1B
3540 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
```


```
3550 LOCATE #3,1,1:INPUT #3,"Load
in picture  Y/N ";pict$
3560 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3570 IF pict$="Y" OR pict$="y" THE
N GOTO 3580:ELSE RETURN
3580 LOCATE #3,1,1:INPUT #3,"Print
 picture name ";picture$
3590 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3600 LOAD picture$,&6675
3610 CLS #3
3620 RETURN
3630 REM go to option pages
3640 CALL &BB1B:IF INKEY(52)=128 T
HEN GOTO 3640:CALL &BB1B
3650 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3660 LOCATE #3,1,1:INPUT #3,"Go to
 options Y/N";yes$:IF yes$="Y" OR
yes$="y" THEN MODE 2:y=y+2: GOTO 2
20: REM go to option pages
3670 LOCATE #3,1,1:PRINT #3,SPACE$
(40)
3680 RETURN
3690 REM ** save picture to memory
 machine code **
3700 FOR n=26203 TO 26226
3710 READ x
3720 POKE n,x
3730 NEXT n
3740 RETURN
3750 DATA 1,0,64,33,0,192,17,117,1
02,237,176,201
3760 DATA 1,0,64,33,117,102,17,0,1
92,237,176,201
```

Listing II

```
10 MODE 0:INK 0,13:INK 15,1
20 LOAD "!filename",&C000
```

Listing III

```
10  MEMORY &665A :REM save memory
for picture and disc drive
20  GOSUB 3700: REM save picture t
o memory machine code
30 LOAD "jet",&6675 :REM load pict
ure into memory
40 MODE 0:INK 14,9:INK 0,13:INK 15
,1:REM set up screen mode and colo
urs
50 CALL 26215 : REM picture to scr
een
60 FOR n= 1 TO 2000:NEXT:REM wait
70 DRAW 640,400,3 :FOR n= 1 TO 100
0:NEXT: REM draw line across pictu
re
80 CALL 26203 : REM save new pictu
re
90 CLS:FOR n= 1 TO 1000:NEXT
100 CALL 26215 :REM new updated pi
cture
110 END
3690 REM ** save picture to memory
 machine code **
3700 FOR n=26203 TO 26226
3710 READ x
3720 POKE n,x
3730 NEXT n
3740 RETURN
3750 DATA 1,0,64,33,0,192,17,117,1
02,237,176,201
3760 DATA 1,0,64,33,117,102,17,0,1
92,237,176,201
```

MAJOR VARIABLES

picture$ Name for screen picture.
movx Cursor step for mode choice.
beam Beam toggle.
beamtrip Trips between B and Copy.
x and y Horizontal and vertical cursor positions.
tes Result of TEST pixel.
jump, jumpy Fast horizontal, vertical cursor move.
xp, yp Tab setting.
fil Draw to Tab toggle.
tb Tab set.
infil Fill toggle.
movetab Tab moving toggle.
extrax, extray Tab move jump.
mem Shape memory toggle.
memtrip Start/stop memory.
radx, rady Horizontal and vertical radii.
stpos Circle start position.
side Number of sides.
delay Delay in polygon.
cw Clockwise or anticlockwise.
startat Start position, on circumference of centre.
tba Draw to Tab.
poly Circumference divided by number of sides.
rstep Cursor moved in pixels (arrow option).
rtes Radius option TEST.
sh, shape Number of shape.
mag Magnification or reduction.
space,minus Positions for blank$(n).
character$(n) Shape string.
blank$(n) Position of blanks or minus in character$.

SUBROUTINES

50 Start, lowers memory and loads machine code.
350 Initialise, sets up the screen windows and colours.
500 Main program.
950 Quick circle.
1030 Pen colour change.
1220 Circles.
1950 Radius.
2050 Type at cursor.
2240 Fill.
2440 Ink change.
2640 Save screen to memory.
2760 Save memory to tape or disc.
2800 Cursor jump.
2920 Draw from memory.
3230 Save shapes.
3350 Input shapes.
3520 Load picture to memory.
3630 Go to option pages.
3690 Machine code, allows saving and recalling a picture to the reserved memory.

SUMMARY OF OPTIONS

Beam ... Ctrl + B
Cursor ... Arrows
Line to Tab ... Shift + Tab
Clear screen ... Shift + Clr
Circle/Polygon ... Ctrl + Clr
Draw to Tab ... Ctrl + D
Fill ... Ctrl + F
Go back to options ... Ctrl + G
Input shape file ... Ctrl + I
Jump cursor ... Ctrl + J
Load screen ... Ctrl + L
Memory shape ... Ctrl + M
Pen ink change ... Ctrl + P
Quick circle ... Ctrl + Q
Redraw ... Ctrl + R
Save shape file ... Ctrl + S
Save screen ... Ctrl + ?
Tab setting ... Tab
Tab move ... Ctrl + Tab
Type at cursor ... Ctrl + T

THINK BIG and get yours NOTICED

Adding a printer to your computer can transform it from an expensive toy into a genuinely useful piece of equipment. For most people printers mean word processing, hard copy from spreadsheets or databases, graphics or program listings.

IAN SHARPE finds Tas–Sign a way to get good notices

Now Tasman, author of one of the two most popular Amstrad word processors and many other printer–related programs, has added another use to the list of possibilities.

In conjunction with your CPC6128 or PCW8256/8512 and dot matrix printer Tas–Sign will enable you to produce sign writing in a variety of styles and sizes. It runs under CP/M plus and is a surprisingly versatile piece of software.

Tas–Sign differs from simple "big character" programs in the range of layout options and the quality of its output. Whatever scale of magnification you apply to the characters, slopes and curves are always drawn at the limit of resolution of your printer.

So a C eight inches high is not composed of an 8 x 8 matrix with inch tall rectangular pixels. Instead it's a smooth curve, and only if you look very closely does the stepping (or aliasing) become visible.

Tasman has achieved this by storing the character shapes as mathematical descriptions rather than matrixes as found in your CPC6128 manual. To look at it another way, a circle drawn using a formula to calculate the position of the points will be smooth if enough points are used, whereas drawing a large circle by simply magnifying one plotted in a matrix will produce a crude, chunky shape.

Four character sets are supplied: Standard, Western, Casual and Block. Casual looks as though it has been produced with a nib or brush and Block is a stencil–style font. Each can be printed in a huge range of sizes with a horizontal stretch factor, either normally or in italics and with optional underlining or inversed colours.

There is no facility to design your own character set. At first this might seem like an omission, but the complex way in which characters are stored would make writing such a utility a far from trivial task.

With this in mind, it's a pity that a set of graphics characters wasn't included as this would have greatly enhanced the package. Space is tight on the disc, but there is room for one more font, so how about it chaps?

Characters are plotted in one of a choice of fill styles, ranging from solid through various combinations of horizontal and diagonal stripes.

The patterns can be printed in single or double strike, single or double density and there's an option to have the print head make an additional pass to fill in the spaces between dots. The fill pattern can also be used to surround the finished article with a border.

The ability to draw outlines of characters, leaving the centres blank, would have been useful, because I think it might be preferable on larger signs to spare your printer ribbon and fill in the shapes with poster paint.

As well as character size, font and fill style, the program offers control over most other aspects of the finished appearance.

There are three spacing options. Characters can be printed at a regular pitch or proportionally, in other words the spacing is adjusted according to the letter involved to give a more even appearance. The latter option can be further refined to include kerning where letters that overlap, such as the combination WA, are placed closer together.

Text can be printed horizontally for smaller signs or vertically for large banners. The gaps between lines are definable and commands to centre text on the page or align it with the right edge complete the picture.

The text of your sign is input row by row in the editing screen, which is shown in Figure I. You don't design your sign on–screen – instead the final appearance is controlled by setting the various options and embedding commands in the text.

The best approach is to have your sign planned on paper with the settings and commands worked out before using the program.

The commands can be referenced either from the manual or by scrolling the Help screen to the appropriate point. As well as setting up the various options in the editing screen, the facility to embed commands can be used to turn underlining or swap to another font or fill style part way through printing the sign.

When you've finished inputting the information you can save it to disc and all that remains is to let Tas–Sign get on with the printing. For all but the smallest pieces this is the walk–away–and–leave–it type operation familiar to rhubarb growers and Mandelbrot set plotters the world over.

Despite Epson control codes being an industry standard, printers do vary and Tasman has provided a selection of configurations for popular printers with the facility to define customised drivers.

The manual runs to 31 pages and includes a tutorial section. It is complete, but I needed to read some parts several times and do some experimenting before I was clear on some points.

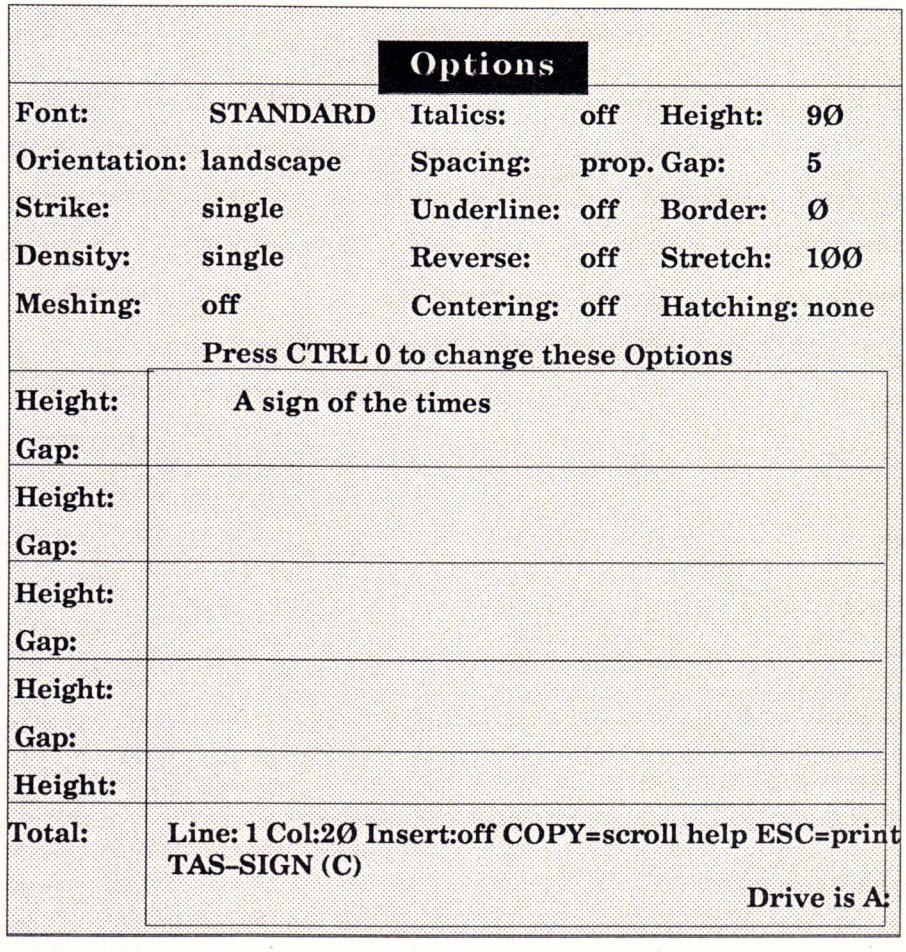


Figure I: The editing screen

There are also several examples on the disc, so by the time you've worked through them you'll have an understanding of the basics. If you have any problems, Tasman has a reputation for being particularly helpful.

Although there is some error–trapping, it has to be said that it's possible to confuse the program. For instance, embedding a command in the top line of text to make some characters taller than the overall setting causes it to lock up. In this case you need to make the top row as tall as the tallest letters and reduce the size where appropriate. Make sure you save your sign before printing it!

Tas–Sign is a very flexible piece of software and because of that I haven't been able to mention all of its many features.

It's probably a little bit pricey for the casual user. But anyone who regularly needs good quality sign writing in small quantities will find this a comprehensively featured piece of software that could earn its keep.

FADE AWAY

ROLAND WADDILOVE starts a new series examining machine code graphics on the PCW

In this short series we'll be developing some simple machine code graphics routines for the PCW. These can be used to enhance your Basic programs and will enable you to produce impressive screen displays and games like CPC owners.

```
        ;— — — — — — — — —
        ;PCW Machine Code Routines
        ;— — — — — — — — —
        ORG 0E000H
        ;— — — — — — — — —
        ;       Fancy CLS
        ;— — — — — — — — —
fade:   CALL USERF
        DI                      ; fast!
        LD B,8
fdi:    LD HL,5930H             ; start
        LD DE,32*720            ;size
fd2:    SLA (HL)                ;rotate
        INC HL                  ;next byte
        DEC DE
        LD A,D
        OR E
        JP NZ,fd2               ;done?
        DJNZ fdl
        E1
        RET
        ;— — — — — — — — —
        ;       Switch on screen
        ;— — — — — — — — —
        USERF: POP BC           ;get PC
        CALL 0FC5AH             ;B10S+30
        DW 0E9H
        RET

        END
```

Listing I

The routines we'll be looking at were originally written for the arcade games Snow Plough and Christmas Box on the Christmas Crackers disc. If you have this take a look and see all the routines in action.

I'll kick off the series with a short, simple routine to clear the screen, though this isn't the usual Mallard Basic method:

PRINT CHR$(27);"E"

It is a fancy version in which the screen display slowly fades away.

To see what I mean enter and run Program I. (Don't forget to save it first!) The screen will fill with text and the program will wait for you to press a key. Tap the spacebar and the characters will melt away.

The Basic listing is fairly straightforward: Line 50 clears the screen and 60 switches off the cursor. The three lines of data at the

continues page 62

```
1Ø REM PCW Screen Fade
2Ø REM By R.A.Waddilove
3Ø REM (c) Computing with the Amstrad
4Ø MEMORY &HDFFF
5Ø PRINT CHR$(27);"E";CHR$(27);"H"
6Ø PRINT CHR$(27);"f"
7Ø address=&HEØØØ
8Ø FOR i=1 TO 3
9Ø sum=Ø:READ code$,check$
1ØØ FOR J=1 21 STEP 2
11Ø byte=VAL("&H"+MID$(code$,j,2))
12Ø POKE address,byte
13Ø sum=sum+byte:address=address+1
14Ø NEXT
15Ø IF sum<>VAL("&H"+check$) THEN
      PRINT "Error in data":END
16Ø NEXT
17Ø:
18Ø FOR i=1 TO 2ØØ
19Ø PRINT"Press a key...";
2ØØ NEXT
21Ø WHILE INKEY$,."": WEND
22Ø WHILE INKEY$="": WEND
23Ø p=&HEØØØ:CALL p
24Ø PRINT CHR$(27);"e"
25Ø END
26Ø :
27Ø DATA CD19EØF3Ø6Ø8213Ø5911
      ØØ,382
28Ø DATA 5ACB26231B7AB3C2ØCE
      Ø1Ø,474
29Ø DATA EFFBC9C1CD5AFCE9ØØC
      9ØØ,749
```

Program I

by Roland Waddilove

CHRISTMAS is here once more and Santa is getting ready to deliver all his presents. The sleigh is waiting and all he needs to do is load up the goodies.

Rudolph has been put out to pasture and the sleigh is now gas powered. It's a bit of a guzzler so take it easy and keep an eye on the fuel gauge.

The presents are scattered throughout Santas's warehouse and you must guide him round on his sleigh and pick them up.

When you have collected all the presents in one room you can move on to the next through the exit at the bottom of the screen.

This is by no means easy since many dangers lurk deep within the warehouse. Watch out for the large spider desceneding on its thread and dodge the Christmas crackers bouncing up and down.

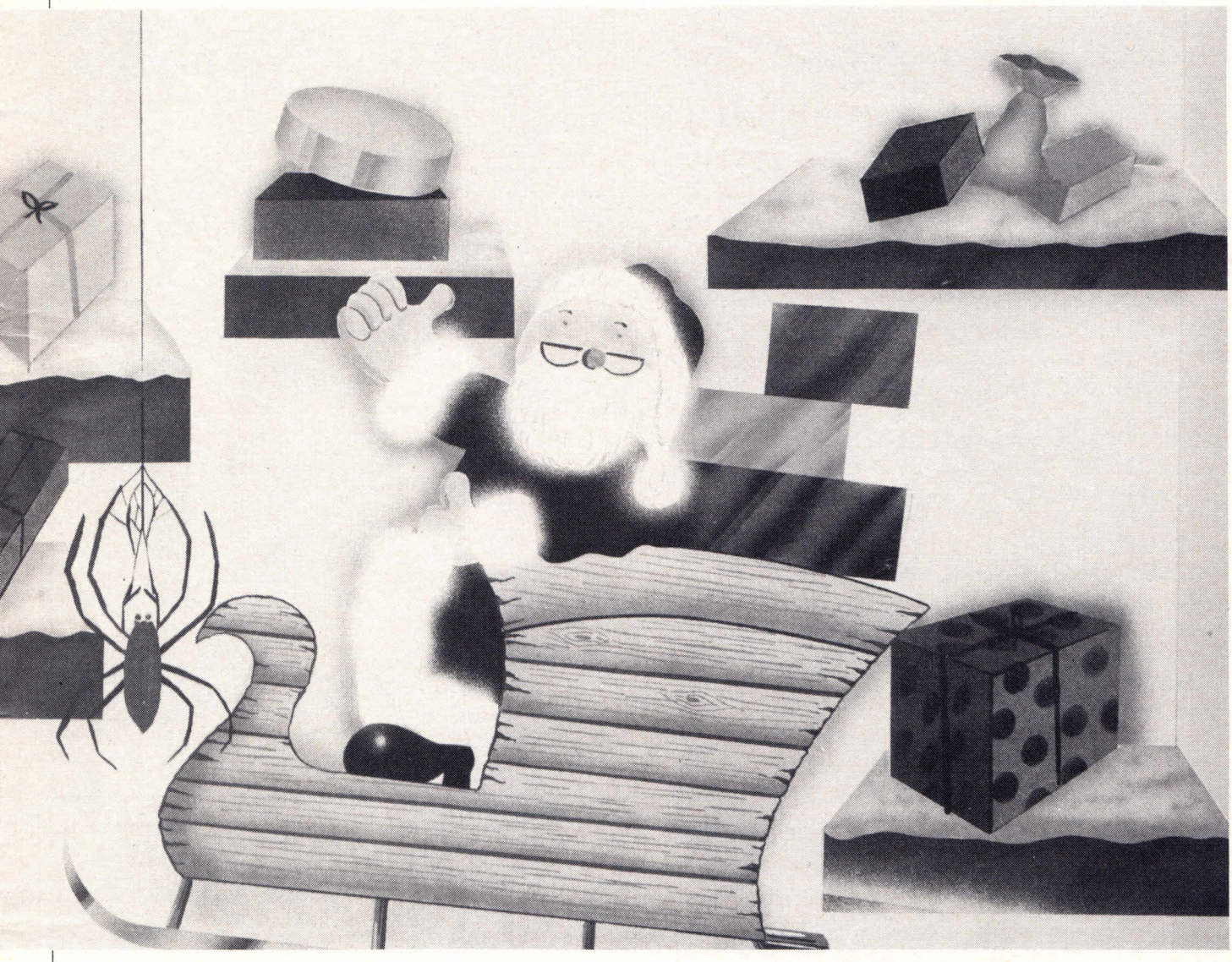
CONTROLS

Shift = thrust
Z = left
X = right

VARIABLES

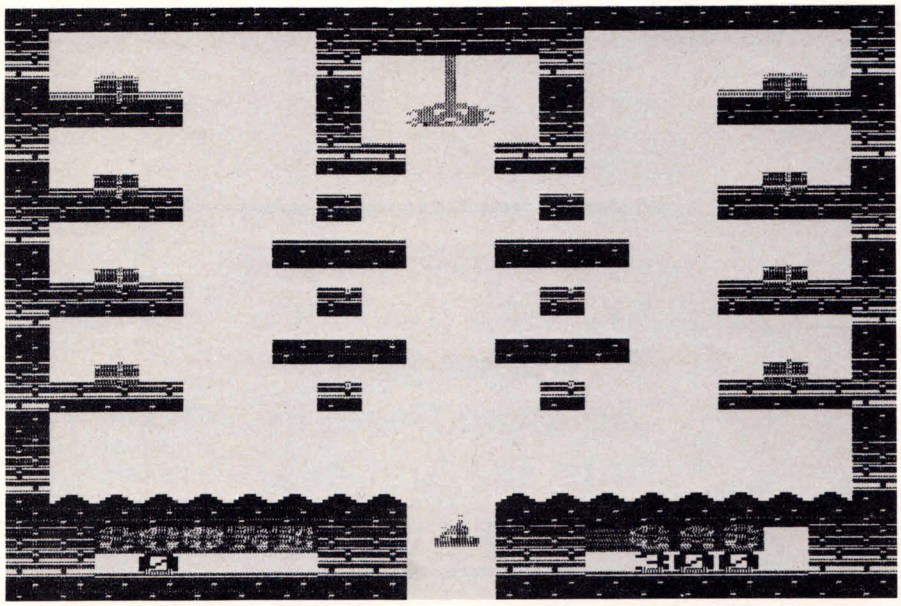
x,y Santa's coordinates
g Gas
s Score
L Presents left
yd Y direction
xd X direction
bug Spider coordinate
cr1 Cracker 1 Y coordinate
cr2 Cracker 2 Y coordinate

```
10 REM ****** Santa's Sleigh *****
20 REM ***** By R.A.Waddilove ****
30 ' (c)Computing with the Amstrad
40 GOSUB 780:REM Initialise
50 GOSUB 280:LOCATE 5,15:PEN 2:PRI
NT"Z=left   X=right   SHIFT=thrust
":LOCATE 10,20:PAPER 1:PEN 3:PRINT
" Press SPACE to play ":WHILE INKE
Y$<>"":WEND:CALL &BB06
60 WHILE 1
70 RESTORE 80:FOR i=0 TO 15:READ j
:INK i,j:NEXT
80 DATA 0,15,18,2,24,8,6,1,0,0,0,0
,20,26,3,24
90 screen=1:lives=3:s=0
100 WHILE lives
110 GOSUB 380:REM Screen
```



```
120 WHILE ok AND y>20
130 GOSUB 970:REM Spider
140 GOSUB 290:REM Santa
150 a=TEST(x+4,y+2):b=TEST(x+28,y+
2):c=TEST(x+28,y-16):d=TEST(x,y-16
)
160 IF c=5 OR d=5 OR TEST(x-4,y-14
)=5 OR TEST(x+32,y-14)=5 THEN GOSU
B 340
170 IF (a>0 AND a<4) OR (b>0 AND b
<4) OR (c>0 AND c<4) OR (d>0 AND d
<4) THEN ok=0
180 WEND
190 IF L=0 THEN screen=screen-(scr
een<3):GOSUB 260 ELSE FOR i=1 TO 1
00:MOVE x+16,y-8:DRAWR INT(RND*100
)-50,INT(RND*50)-25,INT(RND*16):NE
XT:lives=lives-1
200 WEND
210 IF s>hi THEN hi=s
220 GOSUB 280:LOCATE 10,15:PEN 2:P
APER 0:PRINT"Today's high score";h
i:LOCATE 3,20:PAPER 3:PEN 1:PRINT"
 Press SPACE to play another game
":WHILE INKEY$<>"":WEND:CALL &BB06
230 WEND
240 END
250 REM ------ Bonus -----
260 LOCATE 7,10:PEN 4:PAPER 6:PRIN
T" BONUS! ":PAPER 0:FOR i=g TO 0 S
TEP -1:SOUND 1,50,1,13:s=s+1:LOCAT
E 4,24:PRINT MID$(STR$(s),2):LOCAT
E 14,24:PRINT i:NEXT:FOR i=0 TO 20
00:NEXT:RETURN
270 REM ------ Title -----
280 MODE 1:BORDER 0:INK 0,0:INK 1,
```

```
24:INK 2,20:INK 3,6:LOCATE 14,5:PE
N 1:PAPER 0:PRINT"Santa's Sleigh":
LOCATE 13,6:PEN 3:PRINT STRING$(16
,208):RETURN
290 REM ----- Move Santa -----
300 xd=8*((INKEY(71)>-1)-(INKEY(63
)>-1)):IF xd THEN santa=251+SGN(xd
)
310 IF INKEY(21)>-1 THEN SOUND 129
,1000,50,12,0,0,15:yd=yd+1:g=g-1:P
EN 12:LOCATE 14,24:PRINT g ELSE IF
 (TEST(x,y-16)=13 OR TEST(x+32,y-1
6)=13) AND yd<2 THEN yd=0 ELSE yd=
yd-1
320 x=x+xd:y=y+yd:TAG:MOVE x-xd,y-
yd:PRINT" ";:MOVE x,y:PRINT CHR$(s
anta);:TAGOFF
330 RETURN
340 REM ----- Present -----
350 IF x<320 THEN i=3 ELSE i=18
360 SOUND 129,20,100,15,1:LOCATE i
,4*(1+((400-y)\16)\4):PRINT" ":s=s
+10:L=L-1:LOCATE 4,24:PEN 12:PRINT
 MID$(STR$(s),2)
370 RETURN
380 REM ----- Screen -----
390 MODE 0:PEN 4:PAPER 6:LOCATE 6,
10:PRINT" Screen";screen:FOR i=0 T
O 5000:NEXT:PAPER 0:CLS
400 g=300:L=8:RESTORE 580
410 FOR i=1 TO 25:READ a$
420 FOR j=1 TO LEN(a$)
430 PEN 1:PAPER 13:IF a$="*" THEN
PRINT CHR$(240);STRING$(18,9);CHR$
(240);:GOTO 510
440 k=ASC(MID$(a$,j,1))
450 IF k=32 THEN PRINT CHR$(9);
460 IF k=50 THEN PAPER 0:PRINT CHR
$(241);
470 IF k=48 THEN PRINT CHR$(240);
480 IF k=49 THEN PAPER 4:PEN 5:PRI
NT CHR$(242);
490 IF k=51 THEN PEN 13:PAPER 0:PR
INT CHR$(210);
500 NEXT
510 NEXT
520 LOCATE 3,23:PEN 15:PAPER 14:PR
INT"SCORE":LOCATE 14,23:PRINT" GAS
 ":PAPER 0:LOCATE 4,24:PEN 12:PRIN
T MID$(STR$(s),2):LOCATE 14,24:PRI
NT g
530 ok=1:x=310:y=50:xd=0:yd=2:sant
a=250:PLOT -10,-10,6
540 sy=8*screen:sd=1:bug=3:PEN 3:P
APER 0:LOCATE 10,bug:PRINT spiderd
$
550 cr1=5:cr2=15:cd1=1:cd2=-1
560 z=FRE("")
570 RETURN
580 DATA 00000000000000000000
590 DATA 0       000000       0
600 DATA 0       0    0       0
610 DATA 0313    0    0    3130
620 DATA 0000    0    0    0000
630 DATA 0       03  30       0
640 DATA 0       00  00       0
650 DATA 0313                 3130
660 DATA 0000                 0000,*
670 DATA 0      000  000      0
680 DATA 0313                 3130
690 DATA 0000                 0000,*
700 DATA 0      000  000      0
710 DATA 0313                 3130
720 DATA 0000                 0000,*,*,
*
730 DATA 022222222  222222220
740 DATA 000000000  000000000
750 DATA 00      00  00      00
760 DATA 00      00  00      00
770 DATA 000000000  000000000
780 REM ----- Initialise -----
790 DEFINT a-y
800 ENT 1,10,-10,1,127,-5,1:ENV 1,
1,14,1,14,-1,10
810 SYMBOL 240,0,251,251,251,0,223
,223,223
820 SYMBOL 241,0,0,0,0,24,60,126,2
55
830 SYMBOL 242,118,247,247,247,0,2
47,247,247
840 SYMBOL 243,159,95,255,127,127,
255,95,159
850 SYMBOL 244,249,250,255,254,254
,255,250,249
860 SYMBOL 245,7,31,121,187,63,126
,189,95
870 SYMBOL 246,224,248,158,221,252
,126,189,250
880 SYMBOL 247,1,1,1,1,1,1,1,1
890 SYMBOL 248,128,128,128,128,128
,128,128,128
900 SYMBOL 252,16,56,16,60,50,185,
253,255
910 SYMBOL 250,8,28,8,60,76,157,25
5,255
```



```
920 spiderd$=CHR$(247)+CHR$(248)+C
HR$(8)+CHR$(8)+CHR$(10)+CHR$(245)+
CHR$(246)
930 spideru$=CHR$(245)+CHR$(246)+C
HR$(8)+CHR$(8)+CHR$(10)+"  "
940 cr$=CHR$(243)+CHR$(244)
950 BORDER 0:hi=0
960 RETURN
970 REM - Move Spider+Crackers -
980 bug=bug+sd:IF bug=sy OR bug=3
THEN sd=-sd:SOUND 130,2000,100,0,1
,1
990 PEN 3:LOCATE 10,bug:IF sd>0 TH
EN PRINT spiderd$ ELSE PRINT spide
ru$
1000 PEN 2:IF screen<3 THEN FOR i=
0 TO 100:NEXT:RETURN
1010 LOCATE 5,cr1:PRINT "  ":LOCAT
E 5,cr1+cd1:PRINT cr$:cr1=cr1+cd1:
IF cr1=2 OR cr1=20 THEN cd1=-cd1:S
OUND 132,1000,100,0,1,1
1020 LOCATE 15,cr2:PRINT "  ":LOCA
TE 15,cr2+cd2:PRINT cr$:cr2=cr2+cd
2:IF cr2=2 OR cr2=20 THEN cd2=-cd2
:SOUND 132,1000,100,0,1,1
1030 RETURN
```

Computing With The Amstrad

Computing With The Amstrad welcomes program listings and articles for publication. Material should be typed or computer printed, and must be double spaced. Program listings should be accompanied by cassette tape or disk. Please enclose a stamped addressed envelope or the return of material cannot be guaranteed. Contributions accepted for publication by Database Publication or its licensee will be on an all-rights basis.

Computing With The Amstrad is an independent publication and neither *Computing With The Amstrad* and neither Amstrad plc or Amsoft or their distributors are responsible for any of the articles in this issue or for any of the opinions expressed.

How BASIC Works

JOHN HUGHES explains the inner workings of Amstrad Basic in this, the first of a two part series

Fantastic, isn't it? You plug in your Amstrad, flick a couple of switches, and up comes that reassuring message telling you that Basic is at your command.

In this two part series we'll be taking a look at how the Basic works. The general principles are pretty much the same for most other micros as well, but there are just enough differences to make it very interesting – or frustrating – if you try to transfer what you learn here directly to other machines.

Perhaps the most important difference between Basic and other high level computing languages such as Fortran and Cobol is that Basic is an interpreted language, whereas the latter two are compiled.

That is to say, a program written in Cobol or Fortran is turned into machine code en bloc and then executed whereas one written in Basic is turned into machine code statement by statement, each statement being executed as it is interpreted. There are compiled Basics, but they are outside the scope of these articles.

This has various effects, both good and bad. The ones that most concern us here are that it makes Basic rather a slow–running language – although this isn't too apparent on most modern machines unless you go in for a lot of screen animation.

It also means that the source program remains in the computer's memory all the time it is being executed, and that in turn means that it is very easy for us to watch it working.

Now it's time to actually poke around in your Amstrad's innards – or PEEK around, to be more precise. Reset your computer with Ctrl+Shift+Esc then enter the following Basic program:

```
10 REM This is a remark

20 PRINT "This is a PRINT statement"
```

Then type in the following line in direct mode. You will probably find it easier to follow this if you are in Mode 2, as this allows so much more on the screen at a time:

```
FOR J=368 TO 428:PRINT PEEK(J);:NEXT
```

and the computer will respond with:

```
23  0  10  0  197  32  84  104  105  115  32
105  115  32  97  32  114  101  109  97  114
107  0  34  0  20  0  191  32  34  84  104
105  115  32  105  115  32  97  32  80  82  73
78  84  32  115  116  97  116  101  109  101
110  116  34  0  0  0  0  0
```

What you are seeing here is how the two lines of Basic program that you have typed in are stored in the computer's memory. Each number represents what is held in one byte of memory, starting from the point at which your program begins.

An important concept to grasp is that, for hardware reasons, computers store many numbers in what humans would regard as the wrong order – the low number first, and the high number second.

It's a bit like 21 (twenty–one) being written down as 12! In this case though, rather than tens and units the hi and lo "columns" are 256 and units respectively. See our machine code series or Bits and Bytes to see why.

You can see this so–called "lo–byte/hi–byte" situation in the first two numbers – 23,0. These mean that this line of program is 23 bytes long (including the two bytes used for this information.)

The next two numbers, 10,0 are also lo–byte/hi–byte, and tell us the number of the first line of the program.

After this comes the code 197, which stands for the Basic keyword REM – all Basic keywords are represented in a similar way, and the codes are called tokens. Note, by the way, that the space which you typed in after the line number is not represented in the memory.

Then it gets easy. If you look up the next 17 numbers, as far as 107,

in the User Instructions, you will see that they are the Ascii codes for the remaining letters in the statement – 32 is a space, 84 is a capital T and so on.

Finally, the sequence finishes with zero as an end–of–line marker.

All Basic command statements are represented in the same general way. If you're still confused. Figure I may help to make things a little clearer.

The second statement occupies 34 bytes, as you will see from perusing the next two numbers. Then comes the second line number, 20, 0, and the token 191, which stands for PRINT, then a space, 32, a double quote, 34 and so on.

But note that here the word PRINT, which formed part of the message included in the quotation marks, is listed as a series of Ascii values – 80, 82, 73, 78, 84 – rather than as a token, because tokens only stand for instructions that the computer has to obey, and not for similar words which occur inside quotation marks.

As this point you would probably assume that each Basic keyword has its own unique token to go with it, but this is not quite true. If the second line of your original program had been written as:

2Ø? "This is a PRINT statement"

using? as a substitute for PRINT, you would find that the token for? is 191, the same as for the keyword spelled out in full.

This is why when you list a program written using? the computer always replaces it with PRINT – it has no way of knowing which form of the command you originally used.

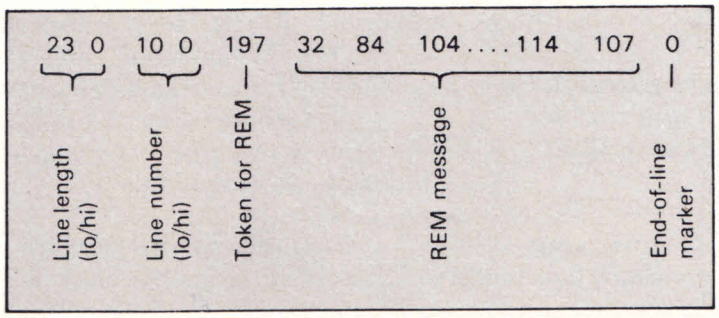


*Figure I: How **10 REM This is a remark** is stored*

But the other obvious equivalence, between REM and', doesn't work in the same way. We've already seen that the token for REM is 197, and you might guess that when use ' as a keyword it too would have the same token.

But in fact the token is 1,192 – so, far from saving memory with the shorter form, it actually takes up another byte!

But because the tokens are different, the Basic interpreter is able to distinguish between them – ' isn't turned into REM in the same way as ? is turned into PRINT when you list a program.

Before you start to see how many tokens you can work out for yourself, there are a couple of curiosities which are worth noticing.

The first is merely a historical oddity, but the second may help you to write programs that are easier to follow when debugging time rolls round.

You probably know that when Basic was first developed it required all assignments to be seen coming – that is, instead of writing A=10, you had to write LET A=10.

Until quite recently some computers still used interpreters which inserted the token for LET (165) even if it didn't appear in the original program.

The Amstrad doesn't put it in unless you have used it yourself, so that you really do save memory space by not using it – which you don't if you use ' and ? instead of REM and PRINT.

The other oddity, which the Amstrad shares with very many other micros, is that anything added to the right–hand end of certain types of statement is ignored by the interpreter *if it is enclosed in brackets.*

This is especially helpful in conditional jumps, where you can insert little mini–REMs after each condition instead of at the end of the whole statement. The following, for example, is certainly not legal in terms of what the User Instructions tell you, but on the other hand it works and is often very useful:

1Ø IF j=i THEN 4Ø (initialise variables) ELSE IF j>1 THEN 1ØØ (main menu)

Next we'll be taking a look at how variables are stored in program lines, and in particular how the Amstrad distinguishes between string and numeric variable types.

PCW MACHINE CODE GRAPHICS

continued from page 55

end of the listing contain the machine code. This is read and stored above HIMEM at &HE000.

Finally the screen is filled with text and the program waits at line 220. The machine code fade routine is called in line 230 then the cursor is switched back on.

Listing I shows the source code for the fade routine written using ZSM a public domain Z80 assembler. If you want to use your PCW to the full I would recommend using Z80 assembly language rather than 8080 as there are many more instructions.

In the December 1986 issue of *Computing with the Amstrad* I briefly introduced graphics with the old classic arcade game Blitz. This article contains useful background information on the PCW's memory map and I'm going to assume you've read this.

The machine code is copied into the middle of the 16k block of common ram at &HE000. This will enable the code to access all the PCW's memory.

The first instruction calls the subroutine USERF to switch in the screen ram. This uses BIOS function call 30 to call an extended BIOS function SCR RUN ROUTINE passing it the address of the fade routine in BC, (popped of the stack).

The main routine uses the B register as a counter while HL is loaded with &H5930, the screen start (not &5940 as I said in the December 1986 article – a typing error!), and DE is loaded with the size, 32 lines each 720 bytes long.

The screen is then cleared by running through the screen ram, shifting the contents of each byte left one bit at a time using SLA (HL). After eight shifts the screen will be clear and the routine ends.

• That's all for now. Next month things will start to get a bit more complicated as we look at some routines for printing large text and user defined graphics characters.

"Here we go again... Catch us if you can!"

These words belong to a number-one hit back in the sixties and are relevant today because they were probably being sung by dealers in the stock and foreign exchange markets over the last few weeks.

As just about everyone reading this will know, the Australian Peso has again been devalued against everything but the Argentinian banana. This of course has serious consequences for those of us who get our fixes from imported software. The immediate effect at the time of writing is a 15% increase across the board on new stocks of all those imported goodies. Keen readers will observe that we have suspended our regular price list until such time as the dollar stops "floating". We have some items at pre-devaluation prices but it would pay you to check before ordering.

The other bit of bad news is the demise of DKTronics in the UK. Apparently the sun shone one day this Northern summer and this caused a drastic fall in the sales of computer peripherals. Fortunately RAM Electronics have stepped into the breach and new products should be available by the time you read this. We understand the major changes in the product line are combined 256K RAM/Silicon Disk Units replacing the previously separate boxes.

Some Good News!

Just when you were beginning to feel really miserable we bring you glad tidings! As mentioned earlier, we have suspended our regular price list but replaced it with another. Our "oldies but goodies" range is not subject to price fluctuation and we have plenty of stock. Check and compare them with what you'd pay for a blank disk. Instant disposable software! If you don't like it - wipe it. Serious users won't want to overlook the bargains in the WP and Spreadsheet departments. Some of these items are selling at a quarter of their original price. Get in quick and fill those stockings. For those who've read this far there's a 10% discount for orders over $100.

With a hint of Christmas in the air we'll take this opportunity to wish all our readers a very Merry Christmas.

$AVINGS ON MAGAZINE$

SPECIAL CHRISTMAS OFFER!

BACK ISSUES – **$3.45** each COMPLETE SET OF 15 ISSUES – **$36**

To take full advantage of this **once only offer** fill in order form on centre page.

GAMES SPECIALS ON DISK

ALL $9.99

(except for one – Fighting Warrior/Exploding Fist – at just $14.99)

Turn to centre pages to order.