



**STOP
PRESS**

DRAGON

EDITORIAL

Welcome to the third edition of Stop Press which, like its predecessors, has been sent to all those Dragon owners who returned their guarantee card. Those who have not (and there are many) are not as yet registered members of Dragon Users Club and will not be on the mailing list. If you know of anybody in this category, gently prod them to return their card so that future issues of Stop Press may be directed to them.

Though not (as yet) a fully-fledged magazine, Stop Press continues, in this and future issues, to offer Dragon owners pages of program-oriented material. Programs which (a) do something interesting (b) illustrate programming techniques and (c) can be typed into Dragon in a short time, are ideal for inclusion in Stop Press.

Most of the material presented falls into at least two of these categories!

Two features in the last issue have their counterparts within, indeed we hope that Maching Code Corner and the Young User Pages will be regular features. In the belief that there are many Dragon users who will appreciate a gentle introduction to machine code, MCC (nothing to do with cricket!) continues the series by showing how the commands introduced last

time can be harnessed to perform high-speed, high-resolution graphics. As each issue is published, readers will be able to accumulate sufficient expertise to make modest use of machine code in their programs.

This does not deny the fact that Dragon's BASIC interpreter offers a rich vocabulary of commands for constructing programs, whether they be games or whatever. Two of the commands, SOUND and PLAY, are explored in the YOUNG USER PAGES which also includes a puzzle and a competition.

Mr W Harold sent a contribution (Patterns) which exploits LINE. He and others will be interested in 'Take 1000 lines' where other intriguing aspects of the LINE command are investigated. Incidentally the pictures in this and the previous issue were not produced by ruler and pencil but by Dragon and a Hewlett Packard graph plotter.

Another contribution, this time from a young user in Cardiff, is a game of skill called 'Chicana'. Full marks to Gareth Rowlands for a superbly designed game and a compact program. We look forward to many other such contributions from young and old alike.

The summer months may see a lull in programmers' activities as they take advantage of the hot

weather! But don't forget to send your entries for the 'Draw a Dragon Logo' competition (see Issue 2) by the end of July please, to the editorial address.

Miss Gethy Hyde,
Dragon Data Ltd., Watling Industrial Estate,
Morpeth, Post Town, West Gosforth

As for Dragon Data, every month, hot weather or otherwise, is a month of intense activity in preparing for the launching of new ventures and the consolidation of software for Dragon 32. Stop Press will play an increasing role in providing up to date information in the months ahead.

PRICE CHANGE

On the 23 May the retail price of Dragon 32 was reduced to £175 including V.A.T. At the same time the following reductions were made: Joysticks selection 2, Graphic Animator, examples from the manual, Dragon Mountain and Dragon Selection 1 down from £7.95 to £4.95; Ghost Attack cartridge down from £24.95 to £19.95. These price adjustments are to be enhanced by the release of approximately 25 new titles over the next few weeks with many more to follow shortly thereafter. Also note that the Dragon disc drive and operating system was launched at the recent Computer Fair at Earls Court, and disc-based software will be released shortly.

MACHINE CODE CORNER



An area of programming in which the effect of machine code is particularly striking is the animation of graphics. To see the limitations of BASIC in this respect, try keying in this short program:

```
10 PMODE4:POLY SCREEN1:COLOR1
20 CIRCLE10,20:3
30 DIMA(10)
40 GET#(0) - (0,0):A
50 FOR I = 1 TO 20:PUT#(0) - (I,0):A:SET NEXT
60 GOTO 30
```

A small circle is drawn on the left of the screen, and moved, by the finest possible increments, to the right. The "animation" takes about 15 seconds. Our more artistic readers may like to convert the circle into a small (by the way, note how small the array dimension is - it needs to be about a fortieth of the area of the "GET" in pixels, which is $21 \times 21 = 441$).

Well, how much can this be improved by using machine code? Try this one:

```
10 DATA 16,57,40,60,30,44,08,76,66,64,1,
   82,3,83
20 DATA 30,30,20,54,20,24,30,19,31,21,16, 8C,1,0,36,0,1,20
30 FOR I = 1 TO 8:READ#(0):POKE I - 3288,VAL"00H" - X:
   NEXT
40 PMODE4:POLY SCREEN1:COLOR1
50 CIRCLE10,20:3
60 EXEC2000
70 GOTO 30
```

This, of course, is really two short programs. Lines 10-30 POKE in the machine code, and lines 40-70 execute it. Even with the READ-POKE loop the program is finished in one second. The action itself takes one fifth of a second. After the first RUN, the machine code will be in position, so RUN40 will give you an action replay.

This is obviously faster than you will need for most purposes, and can be slowed by the use of delay loops. As with the BASIC program, every intermediate position is occupied, so a completely smooth movement is possible at any speed. The BASIC can be speeded up by leaving out some of the intermediate positions, but this results in a jerky movement.

Before we look in detail at animations of this sort, we must investigate the means by which Dragon stores graphic information. We shall concentrate on PMODE4.1. This makes use of memories \$600 to \$10FF (or 1536 to 3678), i.e. a total of 6144 bytes. In this mode, each of the 256x192 pixels is either 'on' (green or buff) or 'off' (black). If it is 'on' it has value 1, if it is 'off' it has value 0. Pixels are

grouped together in sets of eight, so the top line of the screen is made up of 32 sets of 8 pixels, and each set is represented by one byte of memory. For example, if the first eight pixels are 'on, on, off, off, on, on, off, off', then \$600 will contain the binary number 11001100 (decimal 204). This can be seen by running

```
10 PMODE4:SCREEN1:POLY:POKE16,$00:30
20 GOTO 30
```

(POLY turns the whole graphics screen on i.e. it places binary 11111111, or decimal 255, in each memory, \$600 to \$70FF.) So the rows of the screen are contained in memories \$600-\$60F, \$620-\$62F, \$640-\$64F etc.

From this discussion, you may have concluded that it is easier to move things up and down than sideways - and you would be right. The byte configuration remains the same in vertical movement. So let's see if we can make the balloon go up, in a controlled sort of way.

Let's start in BASIC:

```
10 PMODE4:POLY SCREEN1:COLOR1
20 CIRCLE10,100:PPAINT10,100
30 LINE(10,100)-(10,100),
   PSET:LINE(10,100)-(10,100),PSET
40 LINE(10,100)-(10,100),PSET:5
50 DIMA(10):GET#(10,100) - (10,101):A
60 FOR I = 1 TO 10:PUT#(10,100 - I)
   - (10,101):A:SET NEXT
70 GOTO 30
```

Our balloon drawing is (conveniently) confined to the 16th and 17th bytes in rows 100 to 101. As with GET and PUT it is useful to include the bottom blank row to avoid the problem of "rubbing out" the bottom row of the picture as it moves up. Row 100 starts at byte \$600 + $32 \times 163 = \$1A80$, so the top of the picture is \$1A8F and \$1A70, then \$1A8F and \$1A90 etc.

The following assembly language routine will do the job:

			Machine Code	
1	LDY	#1A8F	16 0E 1A 8F	
2	LOOP1	LDAX	7	30 64
3	LOOP1	LDX	3	0C 54
4		STD	-32,X	6D 00 00
5		LDAX	32,X	30 00 20
6		CMPE	#0FFF	10 03 FF FF
7		BNE	LOOP1	20 1C
8		LDAY	-32,Y	31 00 00
9		CMPE	#00F	10 0C 00 FF
10		BNE	LOOP1	20 17
11		RTS		30

One command you may not have met before is Load Effective Address (LEA). The statement LEAX ,Y may be compared with LDX ,Y but instead of loading into X the data indexed by Y, the actual address (the "value" of Y) is loaded into X. The opcodes for LEAX and LEAY are, respectively, hex 32 and hex 31.

The numbers in the operand field, prior to the comma, in lines 4, 5 and 8 are called offsets. They cause the command to be applied to the register value + offset. In other words, line 4 stores the value of D in the memory whose address is 32 less than the value in X, line 5 increments X by 32, line 8 decrements Y by 32.

The byte following an opcode which requires a register as part of its operand is called a postbyte. We met one in the last edition; X+ is represented by postbyte 000. For our present program, we need to know that X and Y are respectively 004 and 0A4 (with no offset) and with offsets in the range -16 to 127 or -128 to -17 the postbytes are 000 and 0A0 (for X and Y), followed by another byte - the offset. Negative offsets are FF = -1, FE = -2, etc.

Our BASIC program, incorporating the machine code, is:

```

10 DATA "B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R",
    "S,T,U,V,W,X,Y,Z"
20 DATA "B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R",
    "S,T,U,V,W,X,Y,Z"
30 FOR I=0TO9:READO:POKE20000
    + I,VAL"0"+O:GOTO 1
40 PMODE4,1:POLY:SCREEN1,0:COLOR0,1
50 CIRCLE(128,128),7:PAINT(128,128)
60 LINE(128,128)-(128,128),PSET:
    LINE(128,128)-(128,128),PSET
70 LINE(128,128)-(128,128),PSET,0
80 EXEC20000
90 GOTO10

```

Finally, to control the speed, we insert between lines 7 and 8 the following delay loop:

	LDX	OFFF	02 7F FC
LOOPS	LEAX	-LC	00 0F
	SNE	LOOPS	20 FC

The postbyte of LEAX - 1,X requires explanation. When the offset lies between -16 and 95, the postbyte represents the offset. For offsets of 0 to 15, postbyte = offset. For offsets of -16 to -1, postbyte = offset + 020.

The only other modification to the machine code is to the relative address of line 10. 20 07 becomes 20 03. The following program gives a controlled movement:

```

10 DATA "B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R",
    "S,T,U,V,W,X,Y,Z"
20 DATA "B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R",
    "S,T,U,V,W,X,Y,Z"
30 FOR I=0TO9:READO:POKE20000 + I,VAL"0"+
    O:GOTO 1
40 PMODE4,1:POLY:SCREEN1,0:COLOR0,1

```

```

50 CIRCLE(128,128),7:PAINT(128,128)
60 LINE(128,128)-(128,128),PSET:
    LINE(128,128)-(128,128),PSET
70 LINE(128,128)-(128,128),PSET,0
80 POKESHTYFF+O:POKEBHTYFF,255
90 EXEC20000
99 GOTO10

```

Line 80 puts the value $5 \times 256 + 255 = 12795$ into memory \$7FFB07FF. Smaller values give a faster movement; larger ones give a slower movement. The result - a very fine control over the speed of the balloon.

PCOPY with Care!

When we write BASIC programs on DRAGON 32 we are well protected by a sophisticated error trap which prevents us from giving commands that cannot be carried out. As an example of this, turn on your Dragon and type PMODE4,5. The answer is FC ERROR - you can't address page 5 in PMODE4 until you have PCLEARED 8 pages (the default is 4). So type PCLEAR8 and then PMODE4,5. This is accepted.

Now type PCLEAR4. Back comes the answer: FC ERROR. You can't PCLEAR down to 4 pages while you are addressing page 5. So type PMODE4,1 and then PCLEAR4. This is accepted.

Most commands are protected in this way. The obvious exception is the POKE command - if you POKE addresses carefully you are likely to lose control - try typing POKE128,2 (128 and 127 control the cursor position).

A less obvious problem is the PCOPY command. You might expect this to be protected in the same way as PMODE and PCLEAR - but it isn't. If you type a program in, and then (without PCLEARING) type PCOPY 1 TO 5, your program will disappear. This is because graphics page 1 has been copied to the location which held your program. Admittedly this would be a silly thing to do but the problem is more likely to arise with programs containing statements like PCOPY 1 TO J, when it is quite easy to get the value of J wrong. So PCOPY with care!

PATTERNS

Thanks to Mr William Harold of Cambridge for this short program. He suggests using densities of between 3 and 6. It is a program that we have seen around in various forms - the interference patterns caused by the discrete nature of the screen pixels are generally referred to as MOIRE patterns. They will occur when lines are plotted close together.

```

10 * PATTERNS
20 * WILLIAM HAROLD 1983
30 CLS:INPUT "ENTER DENSITY":S
40 PMODE4,SCREEN1:POLY
50 FOR I=0 TO 255 STEP 5
60 LINE(I,0)-(I,128),PSET
70 IF I<128 THEN LINE(I,0)-(255,128),PSET
80 NEXT I
90 AS=INKEY:IF AS="" THEN 30 ELSE 30

```

PARTIAL RESTORE



When a program runs on a large number of DATA statements which may have to be accessed in any order, the most convenient method to adopt is to read the whole dataset into an array and use an indexing variable to access it. This takes up a fair amount of room in the machine, however, and an alternative is to read through the whole dataset every time access is required, using the RESTORE command to return to the beginning of the list. This is undoubtedly the most efficient method of conserving memory, but is a rather cumbersome procedure. A partial RESTORE can be achieved using the DATA memory pointer, in addresses 533 and 534 (decimal 5152). By PEELING these addresses at suitable points the first time the data are read, we can store relevant values so that the pointer can be reset to those points by POKING at a later time. The principle is illustrated by this small program.

```
10 FOR I = 10000 TO PEEK(5153) : PEEK(5154)
20 READ R-NEXT
30 CLS:PRINT@;"WHICH STATEMENT?"
40 X = INKEY$:VAL(X) = 0:VAL(X)
 > THEN 40
50 X = VAL(X) : POK(5153, X) : POK(5154, X)
60 READ R:PRINT@;T1;X
70 FOR I = 10000:NEXT:GOTO 10
80 DATA "FIRST STATEMENT"
90 DATA "SECOND STATEMENT"
100 DATA "THIRD STATEMENT"
```

JOYSTICK GAME



This game is for two players (one on each joystick). The left joystick controls the "L" and the right joystick controls the "R". The idea is to manoeuvre your letter about the screen "turning down" as many of the numbers as possible, before your opponent gets to them. You score the value of the number and your score is shown at the top of the screen. If you disappear off one edge of the screen, you will reappear immediately at the opposite edge. The numbers are 1-9 (so if you see an "8", it is just an eight and a seven together). New numbers appear from time to time to keep you busy. The time limit is controlled by the 400 in line 88.

```
10 REM JOYSTICK GAME
20 REM A.D. BAYER 1982
30 CLS:RANDOMIZE
40 L1 = 10:GOSUB 100:J1 = J - 100:
 POK(5153)
50 R1 = 10:GOSUB 100:J2 = J - 100:
 POK(5154)
60 PRINT"LEFT - L":PRINT@;"RIGHT - R"
  :PRINT@;"TIME - T":
70 G1 = 8.571 - G1 - 1
```



Programs

```
10 FOR T = 1 TO 1000: T = OR(T) : S1 = S1 + 1
  : THEN GOTO 10
110 J1 = INT(100 * RND) : GOSUB 200
120 PRINT@;S1;": PRINT@;S2;": PRINT@;T;":
130 NEXT
140 S2 = S1:THEN PRINT@;"RIGHT IS THE
  WINNER":GOTO 10
150 S2 = S1:THEN PRINT@;"LEFT IS THE
  WINNER":GOTO 10
160 PRINT@;"THE RESULT IS A DRAW":
170 PRINT@;"DO YOU WANT ANOTHER GAME?":
180 X = INKEY$:FOR I = 1 TO 1000:
190 X1 = Y:THEN 30
200 X1 = Y:THEN GOTO 100:GOTO 10
210 FOR I = 1 TO 100:GOTO 10
220 FOR X = 1: J1 = 10: NEXT S = S + 1: RETURN
230 J = RND * PEEK(5153) : PEEK
  (5153 + J) = S: THEN GOTO 10: RETURN
240 H = 8: S = JOYSTK(2) * J1 / 10: S: THEN S = -1
250 S = 4: THEN S = 1
260 Y = 8: S = JOYSTK(1) * J1 / 10:
  S = 12: THEN S = -10
270 S = 4: THEN S = 10
280 X = J1 / 10: Y = H * X / 10: S: THEN S = X - 80
290 S = 16: THEN S = S + 80
300 S = J1 - J1 * 10: S = J1 * 10: THEN RETURN
310 PEEK(5153) = S: THEN GOTO 10:
  S1 = PEEK(5153) - 10
320 POK(5153), S1: POK(5154), J1 / 10: S:
  RETURN
```

The program uses the JOYSTK function to check the positions of the joysticks. Some versions of the Dragon primer give the JOYSTK values the wrong way round. The correct version is

```
JOYSTK(0) Right Horizontal
JOYSTK(1) Right Vertical
JOYSTK(2) Left Horizontal
JOYSTK(3) Left Vertical
```

If you want to use the joysticks in a machine code program, the procedure is as follows. Use the JSR instruction to execute the ROM subroutine which begins at address 58112 (decimal 32784), then the relevant values, for the moment the subroutine was executed, will appear in addresses 5154, 5155, 5156 and 5157 (decimal 340-343). This method can, of course, be used from BASIC, using EXEC. These two programs are equivalent: try them with your joysticks.

```
10 PRINT JOYSTK(0);JOYSTK(1);JOYSTK(2);
  JOYSTK(3):GOTO 10
110 REM EXEC:GOTO 58112:PRINT PEEK(5154);PEEK(5155);
  PEEK(5156);GOTO 10
```



The machine code (lines 252 to 262) controls the down-scroll. The following listing may be useful

		Machine Code
	252	0000 00 00 00
	253	0000 00 00 00
254	000	00 00 00
	255	00 00 00
	256	00 00 00
	257	00 00 00
	258	00 00 00
	259	00 00 00
	260	00 00 00
	261	00 00 00
	262	00 00 00

CHICANE

This driving game was sent to us by Gareth Rowlands of Cardiff. A wide road between 2 and 7 may be selected (we suggest you start at 7) and the idea is to stay on the road for as long as possible, using the left and right arrow keys to steer. Those of us who were brought up in a more sedentary age may benefit by editing line 102, to adjust the upper limit from 7 to 9. The benefits of selecting easy options do not last long - the road narrows as you progress.

```

1 REM "Home" G. ROWLANDS
10 CLS:CLG:GOSUB 2000:GOSUB 200
20 AUDIOP=NOTOROFF:M=1
   "INACBCCCCCIBGBC" GOTO 10
30 FOR I=1000 TO P+3:POKE I,200 NEXT
40 FOR I=P+1:LD TO P+80:POKE I,200 NEXT
50 FOR I=P+1+80 TO 3000:POKE I,200 NEXT:RETURN
60 FOR I=1 TO 10: SOUND P*M+25:LD:CC:2000 NEXT:TIMER=0
70 FOR Q=1 TO 10
80 FOR I=1 TO 10:R=R-RND(2):IF R=1 THEN LC=TRUNC(200
   ELSE LC=200:RC=10
90 CP=CP+(RND(200)-220)-(RND(200)-220)
   POKEP-LD,LC:POKEP+80,RC:P=P+(P=3000 AND R=
   1)-P+(3000 AND R=2):GOTO 2000:IF PERIODP=200 THEN
   10 ELSE POKECP,200 NEXT
100 IF P=3000 THEN D=20 ELSE D=0
110 PRINT@INT:TIMER=100/MMS:
   PRINT@D+2;"MILES":SCREEN 0:1 NEXT
120 A=A+10:Z=2:GOSUB 2000:1
130 PRINT@+64;"ROAD":PRINT@D+96;"NARROWS":
140 LD=ERO+EB+A:GOSUB 200:GOSUB 200:GOTO 30
150 T=TIMER:FOR I=1 TO 10:LD:CC:2000:
   POKECP+RND(2)-220:POKECP+RND(2)-220:
   POKECP+RND(2)-200:POKECP+RND(2)-250 NEXT
160 HT=HT+1:IF HT=0
170 PLAY M:M:PRINT@108;"YOU
   SURVIVED"INT(100/M)*MMS:
175 PRINT@208;"WIDTH OF ROAD":SCREEN 1
180 AS=INKEY:IF VAL(AS)=2 OR VAL(AS)=7 THEN 100
190 CLR:LD=ERO+EA+VAL(AS)+A:
   GOSUB 200
200 P=3000:CP=100:GOSUB 200:GOTO 40
210 B=B+5:LD=LD+1:IF B=1 THEN RETURN
220 B=B-5:LD=LD-1:IF B=1 THEN RETURN ELSE 20
230 P=P+1:READ B:IF B="END" THEN RETURN
240 POKE P+2000,VAL(100/M)+R:GOTO 20
250 DATA 80,50,10,5,10
260 DATA 90,60,20,10,20
270 DATA 40,30,10,0,20
280 DATA 70,40,10,0,10
290 DATA 50,50,40,10,0
300 DATA 60,60,30,10,0
310 DATA "END"

```

CODEBREAKER

This is a short program to further illustrate the power of Dragon's string handling capabilities. It translates a typed message into a code which is constructed by randomly permuting the allowed letters. Because the random permutation has been generated using RND(-1) to initialise the random sequence, the code can be deciphered using the same program providing the transmission code is known.

Of course, given time all such codes can be 'cracked' and one of the ways to do that is to use the knowledge gained from 'Letter Count' concerning the frequencies of each of the letters of the alphabet. For a literary reference read 'THE GOLD BUG' in Edgar Allan Poe's 'Tales of Mystery and Suspense'.

```

1 REM CODEBREAKER A.M. SYKES MANS
10 CLS:CLG:GOSUB 2000
   "ABCDEFGHIJKLMNPOQRSTUVWXYZ,!"
20 C0=C1:PRINT C0-C1:PRINT
30 INPUT "FOR CODING, 2 FOR DECODING":C
40 ON C GOSUB 2000:STOP
50 B=0:PRINT:PRINT "YOUR TRANSMISSION
   CODE IS:"
60 GOSUB 2000
70 PRINT@102;"NOW KEY IN YOUR MESSAGE"
80 PRINT " = 20 CHARACTERS LONG AND ENDING
   WITH <ENTER> "
90 GOSUB 2000:GOSUB 2000:CLS
100 PRINT@1;"YOUR MESSAGE IS@":
   GOSUB 2000:PRINT@102:GOSUB 2000:RETURN
110 INPUT "ENTER TRANSMISSION CODE":A
120 GOTO 40
130 Z=RND(-1):FOR I=1 TO B-2:R=0:PRINT
140 C0=C1:FOR I=0 TO 1 STEP -1
150 Z=RND(-1):LEFT=C0,Z-(1)-RIGHT(C0,Z-2)
   +MID(C0,Z,2):NEXT:RETURN
160 B0="" C0=""
170 K0=INKEY:IF K0="" THEN 110
180 C0=INSTR(C0,K0)+C0-INSTR(C0,K0)+IF C0=0 AND
   C0=0 THEN 10
190 IF C0=1 THEN RETURN
200 IF C0=2 THEN L=LEN(M0):M=LEFT(M0,L-1):
   C0=LEFT(M0,L-1):PRINT@L,M0:GOTO 10
210 B0=M0+K0:PRINT@L,M0+K0:GOSUB 2000:GOTO 10
220 R=INSTR(C0,K0)+C0-CM+MID(C0,L,1):RETURN
230 R=INSTR(C0,K0)+C0-CM+MID(C0,L,1):RETURN
240 FOR J=1 TO 200:NEXT:CLS:RETURN

```

LETTER COUNT



This program counts the number of occurrences of letters in writing. Just type in the words and watch the bar chart build up. There is no backspace so type carefully. The program stops when any bar reaches the top. Then you can press ENTER for a table of the number of occurrences of each letter. You can press ENTER for the table at any time during a run but the program is then terminated.

```
1 REM LETTER COUNT M.PEARSON MAY 83
2 CLS:DIMBC(26)
3 PRINT@BC,"ABCDEFGHIJKLMNPOQRSTUVWXYZ";
4 PRINT@BC," 1-";PRINT@BC,"9-";PRINT@BC,"25-";
5 PRINT@BC,"31-";PRINT@BC,"37-";PRINT@BC,"43-";PRINT@BC,"49-";
6
7 IF A < > - 92 THEN 30
8 CLS:FOR I=0 TO 25:PRINT
9 TAB(5):CHR(I+65);:NEXT I:END
10 A=-A+1:GOTO 7
11 L=L+1:IF L=25 THEN A=-MID@A(L,25)
12 IF A<0 THEN 6: THEN 6 ELSE B(A)=B(A)+1
13 I=INT(B(A)/2):IF B(A)/2=I THEN C=C+1:I=I-1: ELSE
14 C=2*I
15 C=C-2:B(A)/2=INT(A/2)
16 PRINT@A:1+2*27*I-8:CHR(I);
17 IF B(A)=25 OR INKEY$=CHR(10) THEN 40 ELSE 10
```

Another way to display the bar chart requires the following changed lines, with line 11b deleted

```
10 CLS:DIMBC(26)
11 I=INT(B(A)/2):IF B(A)/2=I THEN C=C+1:I=I-1: ELSE C=I
```

TAKE 1000 LINES



The LINE command can draw lines on the screen in any direction. Unlike the DRAW command it is not limited to eight directions. The command requires the co-ordinates of the end points of

the line. It is very useful to have the high-resolution grid for reference. Once the end points have been established the LINE command takes another parameter, PSET or PRESET. PSET draws the line in the foreground colour whereas PRESET draws it in the background colour. The colours can be chosen using the COLOR command. This has two parameters FC, the foreground colour and BC, the background. The colours must be chosen with respect to the SCREEN in use. If the COLOR command is omitted the foreground colour will be the highest number available on that screen and the background the lowest. There is another form of the command LINE-(X2,Y2) which draws a line between the new point and the last one used in a LINE command. If the first LINE command in the program is of this form, a line is drawn from the centre of the screen to the point. Using PRESET, the first time round might be one way of removing this line. In addition to PSET and PRESET there is another parameter; by adding B to the LINE command a box is drawn instead of the line, the diagonal of the box being the original line. If BF is added the box is filled in the foreground colour.

The following program shows a triangle subroutine in use. The co-ordinates of the three points are chosen at random and the colours are rotated. To colour the triangles in, a point inside the triangle must be found for the POINT command. In this case the points of the triangle are chosen at random. So which point lies inside the triangle? The point whose co-ordinates are the average of the X values, (XA + XB + XC)/3 and the average of the Y values (YA + YB + YC)/3 will certainly lie inside.

```
1 REM TRIANGLES M.PEARSON MAY 83
2 PMODES:SCREEN=PCRS
3 FOR C=3 TO 4:COLOR C:5
4 XA=INT(255):XB=INT(255):YC=INT(255)
5 XA=INT(255):YB=INT(255):YC=INT(255)
6 GOSUB 7:PRINT@A:XB+XC:3
7 @A:YB+YC:3:END C:;NEXT
8 GOTO 6
9 LINE@A,YA)-(25,25):PSET
10 LINE-(XC,YC):PSET LINE-(XA,YA),
11 PSET:RETURN
```

Writing a lot of LINE commands can be tedious. One way out is to write the command as a subroutine and change the values of the end points either by defining them directly in the program e.g. XA=100, or by reading them from DATA statements.

The following demonstration program shows how DATA statements can be used with a LINE subroutine to make a picture out of filled boxes. The COLOR command allows the boxes to be different colours. Only when a variable is changed does it's new value have to be READ. The program draws ten boxes and then RUN should reveal one very familiar 'box'.

```

1000 REM BOX BY PEARSON MAY 83
1010 PMODE:SCREEN,1:PCLS C1=7:C2=8:C0=C1
1020 DEF PNC(C)= -C-C1:C2-C-C1:C1
1030 FOR I=1 TO 8:READ XA,YA,XB,YB,C:GOSUB 1000:NEXT
1040 FOR I=1 TO 2:READ YA,YB:GOSUB 1000:NEXT
1050 READ XA,XB:GOSUB 1000
1060 FOR I=1 TO 2:READ YA,YB:GOSUB 1000:NEXT

1070 GOTO 1080
1080 COLOR C:G:RND(A,XA)-(XB,YB):PSET,8:RETURN
1090 DATA 20,40,100,120,8,40,60,100,100
      60,100,100,140,7
1100 DATA 100,110,120,100,7,100,60,100,100,7
1110 DATA 70,80,100,60,120,70,80,60,70

```

Another way to change the co-ordinates is in a FOR...NEXT loop. This works well when the co-ordinates change in a regular way. The next few lines use this method to shade the top and sides of the 'box'. Five LINE commands, together with PSET, could have been used but by drawing lines very close together in alternating colours, new colours, not readily available on Dragon, appear. These are created as quite a bit of the way the eye combines colour and the Dragon sets points. The colours are defined in line 10 so that you can experiment with different ones. You can change the step length in lines 70 and 100 and try altering the SCREEN.

Add these lines to the previous program.

```

10 FOR I=1 TO 20 STEP .5:COLOR CO:LINE(100+I,100-I)-(100+I,121-I):PSET
11 CO=PNC(CO):NEXT
12 FOR I=1 TO 20:COLOR CO:LINE (20+I,10-I)-(100+I,10-I):PSET
13 CO=PNC(CO):NEXT

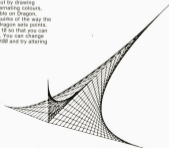
```

There is another sort of shading available with a LINE command. This involves joining points along two lines. Each line is divided into many equal intervals and the points are joined as in peg and silk designs. The effect is of a moulded shape. The following program shows this beautifully.

```

1 REM CONCORDE A MUSEE
10 PMODE:1:COLOR:SCREEN,1:PCLS
20 FOR J=1 TO 3
30 READ N:A:B:C:D:AA:BB:CC:DD:FOR I=0 TO N
40 Z=I/N:X=A+I/C-A/2:I-B+D-DD/2:Y=AA+I/C-AA/2:Y=BB+I/D-BB/2
50 LINE(X,Y)-(X,Y):PSET:NEXT,I
60 GOTO 20
70 DATA 20,10,12,100,100,100,100,100
80 DATA 10,10,12,100,100,100,100,100
90 DATA 20,100,100,100,100,100,100,100

```





YOUNG USER PAGES

NINE-NINE-NINE

Does anyone know what sound a dragon makes? Dogs bark, cows moo, and E.T. says 'gooter', but what do dragons do? The sort of dragons that live in lairs and terrify villagers by burning down houses whenever they cough, that is. Of course the sort that live in houses near the television only terrify the old folks (Over 30's). We know these Dragons sing, make crunch noises and bullet whines. But you haven't really got a well-trained Dragon until it makes noises that you want it to make. You can use the SOUND command to tell your Dragon to make noises. Type in SOUND 89, 10 followed by ENTER. You should hear a middle C lasting about a second. If you don't then try turning up the sound or adjusting the tuning on your TV.

Try using different numbers. The first determines the pitch and can be from 1 to 255. The second number controls the length of the note and must be greater than 0.

If you want to hear all possible pitches you can put the SOUND command into a loop. Going up:-

```

10 FOR P=1 TO 255
20 SOUND P,1
30 NEXT
RUN

```

The loop takes P through all the values from 1 to 255 increasing one at a time. We can change the loop to take P through all values from 255 to 1 decreasing 1 at a time. Just change line 10. Coming down:-

```

10 FOR P= 255 TO 1 STEP -1

```

Perhaps you want to play the same note over and over again. Try:-

```

10 FOR J= 1 TO 5
20 SOUND 198,5
30 NEXT

```

This loop takes J through values 1 to 5 but J isn't mentioned inside the loop so the program just does the same thing five times. How dull! Let's change line 20 to

```

20 SOUND 198,SOUND 198

```

Run this. Does it remind you of anything? Now add these lines to give you the familiar drop in pitch.

```

40 SOUND 198,SOUND 148,5
50 FOR J=1 TO 1
60 SOUND 198,SOUND 148,5
70 NEXT

```

You can make very small changes in pitch using the SOUND command, but you need a very good ear to decide which number gives which note. However with the PLAY command you use letters CDEFGB for a scale together with sharps (#) and flats (-), or alternatively, numbers separated by semi-colons 1,2,3,4, ...11,12 to play successive semitones. This is the way to make your Dragon useful. There are lots of instructions available and page 113 of the Dragon Primer lists them all.

To make the siren sound using the PLAY command I decided to use B and B flat above middle C (B3) played very rapidly (T55) to give the siren effect. I played them 15 times to get the right length of note. For the lower note I used F and F# but you can change the notes to suit yourself. I wanted the siren to start quietly and come nearer so the volume had to be increased. I could write lots of strings: PLAY"BT" + ..._PLAY"Y2" + ... and so on. That's tedious so the changing numbers are in a loop. For example lines 10, 20 and 30 in the following program: are

```

10 FOR L=1 TO 15
20 PLAY"Y" + STR$(L)
30 NEXT L

```

(L for loudness).

When you see several loops in a program it's wise to state which variable you want when you end the loop with NEXT. Hence NEXT L.

Here is the program for a siren getting nearer.

```

10 FOR L=1 TO 15
20 FOR I=1 TO 15
30 PLAY "Y" + STR$(L)
40 PLAY "B#B3" + "-"
50 NEXT I
60 FOR I=1 TO 15
70 PLAY "F#"
80 NEXT I
90 NEXT L

```


If you want to carry on and make the siren go past use

```
## FOR L=20 TO 1 STEP -1
```

and a drop in pitch say A and B flat with F and E.

Now a challenge! Can your Dragon ring like a telephone? If it can, send your program to us and will reward the best entry with free Dragon software! For details, see Young Users Competition.

Now follows a tiny program to develop weird sounds by changing one line.

```
## P1="T2AB8CA"  
## PLAY P1:GO TO ##
```

You must use the BREAK key to stop this program. Here are some other ideas for line 10.

```
## P1="T2M AB8CA"  
## P1="T2M AB8P1"  
## P1="T2M2 1AB8P1P1"  
## P1="T1M2 1AB8P1P1"  
## P1="T1M22 AB8P1P1"  
## P1="T1M2CCCBFFPQCCCBFFP2" (spacecraft sound)  
## P1="T1M2CCCBCCCBFFP2" (sirens ranging fast)  
## P1="T1M2CCCBCCCBFFP2" (sirens slow)  
## P1="T2CCCBCCCBFFP2" (sirens slow)  
## P1="T1M2P2C" (spacehip hi sound)
```

So PLAY AWAY!



WIN 4 SOFTWARE CASSETTES (COMPETITION)

If you are twelve years old or under and would like to win some Dragon Software then devise a program on your Dragon 32 that uses the computer's SOUND or PLAY commands to mimic a telephone ringing. The program should not be more than twenty lines long.

The young programmer judged to have devised the bestest program will win 4 cassettes of their choice from the software listed on the back page and some posters illustrating some popular Dragon games. Send your entries to The Editor of the Newsletter at Dragon Data, by 31st August 1983. The editorial decision is final. The editor, The Newsletter, c/o Dragon Data Ltd, Keating Industrial Estate, Margam, Port Talbot, West Glamorgan.

DRAGON PUZZLE 2



This program plays a tune but first you must get the lines in the correct order. On the right are clues to the missing line numbers. Put the answers in the spaces left for them and type RUN. I'm sure you will recognise the tune.

1 DRAGON PUZZLE 2

```
...PLAY"148524"CB# ... see the start of 'World War 1'  
...PLAY CB#F1=1T02 ... is quite a score  
...PLAY"148524"CB ... is the year Columbus discovered the Bahamas  
...PLAY"14...F ... is a baker's dozen  
...PLAY"148524"CB ... is the year of the Dragon  
...PLAY"148524"CB ... and World War 1 ended  
...PLAY ##+CB#NEXT ... the year Harold saw no more  
...PLAY"148524"CB="148524" ... coming of age
```

SOLUTION TO DRAGON PUZZLE 1

```
## CLS:PRINT"DRAGON PUZZLE"  
## PRINT@20,"FIRE"  
## PRINT@30,"GAMES"  
## PRINT@42,"PIGLET"  
## PRINT@16,"FOOTBALL"  
## PRINT@10,"PYTHON"  
## PRINT@25,"PANDA"  
## PRINT@28,"SCHOOL"  
## PRINT@18,"SNAKE"  
## FOR I=1 TO 3##NEXT  
## FOR I=3 TO 16P=PEZ@162+22P  
## PRINT@27+8CB#P+22  
## NEXT:PRINT@28."
```

(See YOUNG USERS PAGE in previous issue)

**NEW**

RELEASES from the Dragon's Lair

Here are 24 new titles in the official Dragon Data Software range that should be appearing in the shops about now. Watch out for further Software news in STOP PRESS.

M30028 STALAG END

No joysticks. Two adventure games. In stalag, your goal is to escape, alive, from a prisoner of war camp before it is bombed. In End, you are searching for the money left by your late aunt. To prove that you deserve the fortune that has been left to you, your aunt has hidden the cash.

M30024 MANSION OF DOOM

No joysticks. Mansion of doom is an adventure game in which you have been chosen to reunite the crown princess Marlene with the townspeople of her village in Transylvania. You must rescue the princess from the mysterious Count von Steinoff. His mansion, on the edge of the village, is replete with legend. The count himself has never been seen in daylight.

M30023 POSEIDON ADVENTURE

No joysticks. Poseidon Adventure is an adventure game in which you are aboard the luxury liner S.S. Poseidon. A huge undersea earthquake has caused a tidal wave which has capsized the ship. It is floating, bottom-up on the surface and taking on water. Your goal is simply to get out alive.

M30033 DREAMBUG

No joysticks. Dreambug is a monitor (debugger) and disassembler for the Dragon 32 which is designed to be used in conjunction with dream (editor/assembler) cassette for the Dragon 32.

M30023 FINAL COUNTDOWN

No joysticks. Final Countdown is an adventure game in which your mission is to prevent a nuclear missile from being launched and starting world war III. You begin the game outside a missile base which has been evacuated after a berserk General has started the countdown on a missile.

M30030 TIMSCRIPT

No joysticks. Printer required. A speed writer which will enable you to use your Dragon to produce continuous text, such as business letters, quickly and easily. Timscript allows frequently used words and phrases to be replaced by two letter codes, considerably reducing the number of keystrokes required. The codes are automatically expanded when they are typed.

L30010 DREAM

No joysticks. A professional quality screen editor and assembler for use in the production of assembly language programs and subroutines.

D30001 ALLDREAM EDITOR/ASSEMBLER

No joysticks. A screen editor, assembler, disassembler and monitor together on a cartridge. The package will enable the user to create and debug assembly language programs or assembly language subroutines to be called from within basic programs.

L30520 PIXEL EDITOR

No joysticks. This program will enable quick and easy creation and editing of graphics images. Each individual pixel can be accessed in order to produce detailed pictures and character sets.

L30521 HIDE AND SEEK

No joysticks. Hide and seek is a program designed by experts to aid short term memory and develop early reading skills. The program has a range of difficulty levels to suit children between the ages of four and eleven, and, as well as being educational, is a program which children will enjoy using.

L30519 NUMBER PUZZLER

No joysticks. Number puzzler is a program designed by experts to develop childrens powers of mental arithmetic. The program has a range of difficulty levels to suit children between the ages of four and eleven and, as well as being educational is a program which children will enjoy.

L30517 NUMBER GALPER

Joysticks optional. As for number puzzler.

M00528 CIRCUS ADVENTURE

No joysticks. Circus adventure is an adventure game designed especially for children between the ages of 4 and 8. The program incorporates a number of user inputs to encourage keyboard familiarity and presents the child with a series of choices to be made to encourage decision making skills. The program is designed to be non-frustrating and children will enjoy using it.

M30529 SCHOOL MAZE

No joysticks. As for circus adventure.

M00510 DRAGON SELECTION 3

Joysticks required for aliens. Dragon selection 3 is a collection of four games programs- Money, Detective, Aliens, Hangman.

M00511 DRAGON SELECTION 4

No joysticks. Dragon selection 4 is a collection of three party games for children. Password, Lucky Dig, Composer.

J20111 RAIL RUNNER

No joysticks. A fast moving video game, rail runner is a race against time to cross a series of tracks, avoiding trains and handcars to rescue Herman Hobo.

M00518 EL DIABlero

No joysticks. El Diablero is an adult adventure game. You are wandering in the desert trying to regain your lost magic powers before confronting El Diablero.

M00527 STORM ARROWS

Joysticks required. Storm arrows is a fast moving, multi-screen maze game. Your task is to defend yourself against hostile arrows while maintaining your energy supply.

K30112 GALAX ATTACK

Joysticks required. As wave after wave of enemy ships attack, you must defend your ground base against them. The enemy craft fly in convoy formation but they will disband in order to attack.

K30533 SHARK TREASURE

Joysticks required. You have discovered the lost wreck of a ship which sank many years ago with a cargo of gold bars. The only thing between you and a fortune is a swarm of huge sharks. Shark treasure is graphic video game.

J30110 DOODLE BUG

Joysticks required. Doodle bug is a colourful, fast action maze game. An increasing number of enemies have to be avoided together with a number of dangerous obstacles.

K30535 SHUTTLEZAP

Joysticks required. Orbiting the earth your task is to grab as many satellites as you can, before landing safely back at base. This program has great graphics and what's more - it talks to you.

K30114 WHIRLYBIRD RUN

Joysticks required. Whirlybird run is a fast moving arcade type game, with multiple screens, different types of attacker and a maze to be negotiated in the final stages.

DRAGON SOFTWARE



SOFTWARE AVAILABLE FOR THE DRAGON 32

N.3000 DRAGON SELECTION ONE

Four games for the younger user. Written in BASIC, they can be listed and edited.

N.3001 DRAGON SELECTION TWO

Collection of utilities. Create your own data base, write your own tunes.

M.3002 QUEST

Adventure game in medieval setting. Defeat Moorlock, master of the dark castle.

N.3003 MADNESS AND THE MINOTAUR

A real-time adult adventure game.

N.3004 PERSONAL FINANCE

Keep track of family finances.

N.3005 GRAPHIC ANIMATOR

Create simple cartoons on the screen and animate them by flipping through the pages. Joysticks required.

M.3006 COMPUTAVOICE

Your Dragon will talk with this voice synthesiser.

N.3007 EXAMPLES FROM THE MANUAL

30 examples from the programming manual.

N.3008 CALIXTO ISLAND

An adventure game. Return the hidden treasure to its rightful place.

N.3009 BLACK SANCTUM

An adventure game. Overcome the forces of black magic.

M.3010 TYPING TUTOR

Improve your speech and accuracy.

N.3011 DRAGON MOUNTAIN

An adventure game. Defeat the guardians of the treasure hidden in the mountain.

M.3014 FLAG

Place your opponent through a constantly changing maze to the final flag. Joysticks required.

N.3016 EL DABLERO

An adventure game set in the desert.

J.3018 BIPPERK

A challenging shooting game, based on the popular arcade game, one or two players. A high resolution game in black and white. Joysticks required.

J.3019 METEORIDS

Guide your ship through treacherous asteroid belt. A game requiring skill, fast reactions and concentration. A high resolution game in black and white. Joysticks optional.

J.3020 COSMIC INVADERS

Dragon version of the famous arcade game.

J.3021 GHOST ATTACK

Maze game for one player. Joysticks required.

J.3024 CAVE HUNTER

Descend into the maze of caves in search of gold. Joysticks required.

J.3026 STARSHIP CHAMBERLAIN

Protect your planet from the attacks of the Galaxians. High quality arcade game with superb graphics and sound. Joysticks required.

J.3027 ASTROBLAST

Defend your ship against waves of attackers. A high resolution game in black and white. Joysticks required.

L.3028 CHESS

Nine levels of play, from beginner to master.

J.3031 RAIL RUNNER

Move Bill Swishman across the tracks, avoiding trains, to rescue Herman Hobo.