Making the Most of your Atari

Paul Bunn

Foreword by Tim Hartnell

TOTAL CONTROL:

MAKING THE MOST OF YOUR ATARI

PAUL BUNN

Interface 'Success in the Fast Lane' programming series
Foreword by Tim Hartnell

INTERFACE
PUBLICATIONS
44-46 Earls Court Road, London W8 6EJ

‘This Book is dedicated to Ian Nicol’

First published Interface Publications, 1983
Copyright © Paul Bunn, 1983

Any enquiries regarding the contents of this book should be directed by mail to Interface Publications, 44-46 Earls Court Road, London, W6 6EJ

ISBN 0 907563 21 X

First printing, January, 1983

Printed in England by J. W. Dunn (Printers) Limited, Sutton, Surrey.

Contents:

Foreword

Here, in this small volume, in easy-to-understand form, Paul Bunn has brought together all the essential information you'll need to improve your programming techniques on the Atari computers.

From making the most of the graphics, to using sound and the joysticks or paddle, you'll find the vital addresses, the important locations, the programming tricks to get your Atari to do just about anything you want it to do.

And if you just want a generous collection of programs, in ready-to-run form, you'll also find them in this book. From BEETLE JUICE (you, as a small, red beetle, try to cross a busy street without being squashed) to DODGE 'EM (in which you use a joystick to drive your car around a maze, and avoid the computer's car), there are programs for every taste and occasion.

If you've come to the 'What do I do now?' stage with your Atari 400 or 800, then you're ready for Paul Bunn, and this book.

TIM HARTNELL

Author of *THE PERSONAL COMPUTER GUIDE, THE ZX SPECTRUM EXPLORED* and *49 EXPLOSIVE GAMES FOR THE ZX81*

Chapter 1 : INTRODUCTION

This book is aimed at the computer user who has had his Atari computer for one or two months, is happily conversant with Atari Basic but wants to know his machine inside out. This book will provide that and a little bit extra.

Mastering the Atari computer is as simple as mastering the three custom-built chips inside the computer. These chips are called ANTIC, POKEY and GTIA. All these chips are roughly as big as the 6502 microprocessor (in silicon area) giving three times as much computing power as the 6502 alone. These chips relieve the 6502 to do all the computing, leaving the burden of graphics to fall on ANTIC and GTIA. The chip called POKEY handles all input/output including sound generation.

I think that the combination of these four chips makes the Atari one of the best computers on the market today. My decision is based on the high resolution graphics, player-missile graphics, four voice sound playable through the television speaker, being able to have mixed graphics modes and the availability of the three GTIA graphics modes 9 10 and 11 which offer 16 colors on the screen in high resolution graphics.

If all this sounds like double-dutch to you now, do not worry, all will become crystal clear by the time you have finished reading this book.

I am going to describe in detail how to use graphics including the GTIA graphics modes, all the input/output features of the Atari, the error reporting system, player-missile graphics, how to re-define the character set, how to design your own graphics modes, how to read the yellow consol keys, to read the games controllers, how to fully use the sound command and at the end of the book are some games programs and details of how they work.

Chapter 2 :
BASICS OF GRAPHICS

This chapter covers the basics of using the Atari's capabilities of high resolution graphics. Here is a list of the commands associated with graphics:

COMMAND	ABBREVIATION	SHORT DESCRIPTION ON THE FUNCTION OF THE COMMAND
Graphics	GR.	This selects the current graphics mode.
Setcolor	SE.	Re-defines colors from the default values.
Color	C.	Sets color for PLOT or DRAWTO
Plot	PL.	Plots a single point.
Drawto	DR.	Draws a line from the last PLOTed point to the point to the point specified.
Position	POS.	Moves the invisible graphics cursor to the point specified.
Locate	LOC.	Used to check the color of a certain point.
X10 (special fill)	X.	Used in graphics to fill a section or shape.

A more detailed description follows:–

GRAPHICS

This command is used to select one of the various graphics modes. Adding 16 to the number gives a full screen, adding 32 prevents the screen from being cleared when the graphics command was entered. With graphics modes 9 10 and 11, the screen is automatically cleared when any text is printed (including any messages), even when the computer attempts to print READY. This can be overcome by entering the command POKE 703,4 after the GRAPHICS command. This will prevent any text from being printed. Unfortunately, if there is an error in the program, then you will not know what it is. This can be overcome by using this command with the error trapping program described in chapter 5. This program, instead of printing for example ERROR–6 AT LINE 80 would produce–Out of data at line 80. It also goes into graphics mode O (ordinary text mode), which clears the screen and sets all the colors back to the default conditions.

SETCOLOR

This command is used to re-define any of the color registers from their default values. The SETCOLOR command is used as follows:–

SETCOLOR Color register, Color Value, Luminosity

Where the luminosity is an even number ranging from 0–14, where 0 is very dark and 14 very bright, a number of 6 or 8 is a good compromise and goes quite well with most backgrounds.

Color Value	*True Color*	*Color Value*	*True Value*
0	Grey/black	8	Deep blue
1	Light orange	9	Light blue
2	Orange	10	Turquoise
3	Red	11	Green/blue
4	Pink	12	Green
5	Purple	13	Yellow
6	Blue/purple	14	Orange/green
7	Blue	15	Light orange

Memory can be saved by directly POKEing the correct value into the color registers. An example for SETCOLOR 2,3,4 would be POKE 710,3*16+4 or in general POKE color register, color value* 16+ Luminance. "What are the color registers?" I hear you say. A list of them is given below:–

Color Register	*Color Value*	*SETCOLOR Value*
708	1	0
709	2	1
710	3	2
711	4	3

Note that no SETCOLOR command is needed if you wish to use the default colors.

COLOR

This is a command to choose the color that will be used when a DRAWTO or PLOT statement is incurred. It is a little difficult to use the color statement at first because the SETCOLOR command does not use the same numbers as the COLOR statement. Experimenting should eventually give the desired results.

PLOT

This will produce a single pixel at the point specified. Example: PLOT 20,15 will produce a single point at 20,15 or 20 along and down 15. The color will be the last color that was specified. Usually this command is used in conjunction with the DRAWTO command.

DRAWTO

This will produce a line from the last point plotted to the co-ordinates specified, in the color last specified. A small program demonstating the GRAPHICS, COLOR, PLOT, SETCOLOR and DRAWTO commands, is listed below:

```
10 REM ** DEMONSTRATION PROGRAM **
20 REM ** SHOWING BASICS OF     **
30 REM ** GRAPHICS..            **
40 DEG
50 GRAPHICS 7+16
60 SETCOLOR 0,RND(0)*15+1,8
70 SETCOLOR 1,RND(0)*15+1,8
80 SETCOLOR 2,RND(0)*15+1,8
90 COLOR INT(RND(0)*3)+1:SIZE=(RND(0)*10
)+5
100 DX=(RND(0)*120)+20
110 DY=(RND(0)*40)+20
120 FOR R=0 TO 359 STEP 30
130 X=(SIN(R)*SIZE)
140 Y=(COS(R)*SIZE)
150 PLOT DX,DY:DRAWTO DX+X,DY+Y
160 NEXT R
170 GOTO 90
```

Explanation of program:
Line 40: Set all calculations to be in DEGrees.
Line 60–80: Set the color registers 0–2 to random colors with a luminance of 8.
Line 90: Chose the plotting color of one of the randomly picked setcolor values.
Line 100–110: Chose where to position the star, x-y co-ordinates.
Line 120: Set up FOR . . . NEXT loop for drawing star.
Line 130–140: Procedure for working out the position from the co-ordinates where to plot.
Line 150: Plot a single point at the centre of the star, and from there draw to the outside.
Line 160: Continues the FOR + NEXT loop.
Line 170: Makes the program an endless loop so that the program does another star.

POSITION

This command simply moves the invisible graphics cursor to the point specified. This command is useful in the text modes. A program which follows shows the usefulness of this:–

```
10 REM ** POSITION DEMONSTRATION **
20 GRAPHICS 0
30 POKE 752,1
40 X=INT(RND(0)*34)
50 Y=INT(RND(0)*23)
60 POSITION X,Y
70 PRINT "ATARI";
80 GOTO 40
```

An error will occur if you try and position outside the restricted range for the particular graphics mode. This command is more useful in graphics modes 1 and 2, where large text can be printed.

LOCATE

This is another useful command that checks the color of a certain point. The format is like LOCATE X,Y,D. Where X and Y are the X-Y co-ordinates and D is the variable where the data is stored. It will be a zero if it is the background color, and one if it is color 1 and so on.

X10 (SPECIAL FILL)

This command can be used to fill in shapes in a certain color. It is very useful but a little tricky to use at first. A program which draws squares of different sizes and colors is given below. An explanation of how it works is also given:

```
10 REM XIO DEMONSTRATION - SQUARES
20 GRAPHICS 7+16
30 LET COLOUR=INT(RND(0)*3)+1
40 COLOR COLOUR
50 SIZE=INT(RND(0)*10)+5
60 X=(RND(0)*130)+15
70 Y=(RND(0)*66)+15
80 PLOT X+SIZE,Y
90 DRAWTO X+SIZE,Y-SIZE
100 DRAWTO X,Y-SIZE
110 POSITION X,Y
120 POKE 765,COLOUR
130 XIO 18,#6,0,0,"S:"
140 GOTO 30
```

Line 20:	Puts computer in the required graphics mode.
Line 30:	Select a random color.
Line 40:	Set the color to the random value.
Line 50:	Choose a random size.
Line 60,70:	Choose random co-ordinates
Line 80:	PLOT bottom right hand corner.
Line 90:	DRAWTO top right hand corner.
Line 100:	DRAWTO top left hand corner.
Line 110:	Position bottom left hand corner.
Line 120:	POKE location 765 with the color that the fill is to use.
Line 130:	The X10 command that makes the special FILL take place.
Line 140:	Repeats the X10 procedure

GTIA GRAPHICS MODES

All the Atari computers in England are equipped with GTIA graphics modes and with a minimum of 16K memory. I believe that in America the minimum amount of memory is 8K and that GTIA graphics modes are only available if it is upgraded to CTIA.

The GTIA graphics modes all take up to 8K of memory, even more than graphics mode 8. But their graphics capability is enormous. They can provide up to 16 colors on the screen! All modes have a resolution of 80 × 192.

GRAPHICS MODE 9

This graphics mode gives 16 colors all of the same color but at 16 different luminances. The background color sets the color of the background and of the colors with the varying luminances. A color statement from 0 to 15 may be used where 0 is background and 15 is near white. An excellent program which demonstrates the capabilities of graphics mode 9 is as follows. If you haven't seen it before, type it in, type RUN and see the display.

```
10 REM ** GTIA MODE NINE DEMO. **
20 GRAPHICS 9
30 SETCOLOR 4,6,0:Z=16:DEG :X=15
40 Z=Z-1:COLOR Z:I=1
50 FOR P=0 TO 360 STEP 20
60 IF I THEN PLOT SIN(P)*X+40,COS(P)*40+
96:I=0:NEXT P
70 DRAWTO SIN(P)*X+40,COS(P)*40+96
80 NEXT P
90 X=X-1
100 IF X<0 THEN 120
110 GOTO 40
120 GOTO 120
```

GRAPHICS MODE 10

I personally think this is the best graphics mode offered by GTIA. It offers 9 completely different colors, one background and 8 plotting colors. Setting the color registers is easy–just poke the color registers with the color value multiplied by 16 + luminance required. POKE 704 with the background color and POKE 705–712 with colors desired for colors 1 to 8. A program called "Tennis Player" plots a tennis player in graphics mode 10. If you can be bothered to type all that data in, then it will astound you and your friends.

```
10 REM * A tennis player - Paul Bunn *
20 GRAPHICS 10:POKE 704,12*16+2:TRAP 120
30 POKE 705,0:POKE 706,13*16+12:POKE 707
,13*16+8:POKE 708,14*16+8
40 POKE 709,14*16+6
50 POKE 710,4:POKE 711,3*16+4
60 X=0
70 READ A,B,C
80 COLOR B:PLOT X,G:X=X+A:DRAWTO X,G:X=X
-A
90 PLOT X,G+1:X=X+A:DRAWTO X,G+1
100 IF C=1 THEN X=0:G=G+2
110 GOTO 70
120 GOTO 120
130 DATA 39,0,0,7,1,1
140 DATA 37,0,0,10,1,1
150 DATA 36,0,0,12,1,1
160 DATA 35,0,0,14,1,1
170 DATA 34,0,0,15,1,1
180 DATA 34,0,0,9,1,0,1,4,0,5,1,1
190 DATA 34,0,0,5,1,0,5,4,0,6,1,1
200 DATA 34,0,0,2,1,0,9,4,0,5,1,1
210 DATA 34,0,0,1,1,0,10,4,0,3,5,0,1,1,1
220 DATA 35,0,0,6,4,0,2,1,0,2,4,0,1,5,0,
1,1,0,1,5,0,1,1,1
230 DATA 36,0,0,2,1,0,2,4,0,2,1,0,3,4,0,
1,5,0,1,1,0,1,5,0,1,1,1
240 DATA 36,0,0,2,1,0,2,4,0,2,1,0,3,4,0,
3,5,1
250 DATA 36,0,0,2,5,0,6,4,0,3,5,1
260 DATA 36,0,0,2,5,0,6,4,0,3,5,0,8,2,1
270 DATA 37,0,0,1,5,0,6,4,0,4,5,0,9,2,1
280 DATA 37,0,0,1,5,0,1,1,0,1,4,0,2,1,0,
1,4,0,5,5,0,11,2,1
290 DATA 38,0,0,3,5,0,2,4,0,5,5,0,13,2,1
300 DATA 37,0,0,1,2,0,5,4,0,5,5,0,14,2,1
310 DATA 35,0,0,4,2,0,6,4,0,3,5,0,16,2,1
320 DATA 32,0,0,9,2,0,4,4,0,3,5,0,17,2,1
330 DATA 31,0,0,10,2,0,4,4,0,3,5,0,17,2,
0,2,4,1
340 DATA 29,0,0,13,2,0,3,4,0,2,5,0,11,2,
0,1,3,0,5,2,0,4,4,1
350 DATA 28,0,0,16,2,0,2,4,0,1,5,0,11,2,
0,3,3,0,4,2,0,4,4,1
360 DATA 27,0,0,18,2,0,1,4,0,12,2,0,4,3,
0,3,2,0,5,4,1
370 DATA 27,0,0,31,2,0,6,3,0,1,3,0,5,4,1
380 DATA 26,0,0,31,2,0,1,3,0,5,0,0,2,5,0
,5,4,1
390 DATA 25,0,0,32,2,0,1,3,0,6,0,0,1,5,0
,6,4,1
400 DATA 24,0,0,33,2,0,7,0,0,2,5,0,5,4,1
410 DATA 24,0,0,9,2,0,1,3,0,22,2,0,8,0,0
,2,5,0,5,4,1
420 DATA 23,0,0,9,2,0,3,3,0,20,2,0,1,3,0
,9,0,0,1,5,0,5,4,1
430 DATA 23,0,0,8,2,0,1,3,0,1,0,0,2,3,0,
19,2,0,2,3,0,9,0,0,2,5,0,5,4,1
440 DATA 22,0,0,9,2,0,3,0,0,2,3,0,16,2,0
,3,3,0,11,0,0,1,5,0,5,4,1
450 DATA 21,0,0,2,4,0,5,2,0,2,3,0,3,0,0,
4,3,0,11,2,0,6,3,0,13,0,0,1,5,0,4,4,1
460 DATA 20,0,0,4,4,0,1,2,0,4,3,0,4,0,0,
1,2,0,8,3,0,5,2,0,7,3,0,14,0,0,1,5,0,4,4
,1
470 DATA 18,0,0,6,4,0,1,5,0,3,3,0,4,0,0,
1,3,0,3,2,0,3,3,0,9,2,0,6,3,0,14,0,0,1,5
,0,4,4,1
480 DATA 17,0,0,6,4,0,3,5,0,6,0,0,2,3,0,
15,2,0,4,3,0,16,0,0,4,4,1
490 DATA 15,0,0,8,4,0,2,5,0,7,0,0,3,3,0,
15,2,0,3,3,0,17,0,0,4,4,1
500 DATA 14,0,0,8,4,0,2,5,0,7,0,0,5,3,0,
14,2,0,2,3,0,18,0,0,4,4,1
510 DATA 12,0,0,9,4,0,2,5,0,6,0,0,3,2,0,
4,3,0,14,2,0,2,3,0,18,0,0,6,4,1
520 DATA 11,0,0,9,4,0,2,5,0,6,0,0,5,2,0,
4,3,0,12,2,0,3,3,0,18,0,0,7,4,1
530 DATA 11,1,0,9,4,0,1,5,0,7,1,0,5,2,0,
4,3,0,10,2,0,4,3,0,2,2,0,17,1,0,8,4,0,1,
1,1
540 DATA 10,0,0,8,4,0,2,5,0,7,0,0,7,2,0,
4,3,0,8,2,0,3,3,0,4,2,0,18,0,0,2,4,0,1,0
,0,4,4,1
550 DATA 9,1,0,7,4,0,2,5,0,8,1,0,9,2,0,3
,3,0,7,2,0,2,3,0,6,2,0,18,1,0,2,4,0,2,1,
0,4,4,1
560 DATA 8,1,0,8,4,0,10,1,0,10,2,0,3,3,0
,4,2,0,2,3,0,8,2,0,19,1,0,1,4,0,4,1,0,2,
4,1
570 DATA 7,0,0,7,4,0,11,0,0,13,2,0,3,3,0
,12,2,0,19,0,0,1,4,0,4,0,0,2,4,1
580 DATA 7,1,0,1,2,0,3,4,0,1,5,0,13,1,0,
3,4,0,25,2,0,24,1,0,2,4,1
590 DATA 6,1,0,3,2,0,2,5,0,13,1,0,5,4,0,
24,2,0,26,1,1
600 DATA 3,0,0,3,4,0,4,2,0,13,0,0,7,4,0,
22,2,1
610 DATA 1,1,0,6,4,0,2,2,0,14,1,0,8,4,0,
12,2,0,3,4,0,2,5,0,4,2,0,27,1,1
620 DATA 1,1,0,7,4,0,6,1,0,4,7,0,4,1,0,1
0,4,0,8,2,0,7,4,0,4,5,0,1,2,0,27,1,1
630 DATA 1,0,0,3,4,0,2,5,0,2,4,0,4,0,0,7
,7,0,3,0,0,10,4,0,7,2,0,8,4,0,4,5,1
640 DATA 4,4,0,1,5,0,2,6,0,3,1,0,3,7,0,5
,1,0,1,7,0,2,1,0,12,4,0,5,2,0,10,4,0,3,5
,0,28,1,1
650 DATA 5,4,0,3,6,0,3,7,0,7,1,0,1,7,0,2
,1,0,12,4,0,4,2,0,11,4,0,3,5,0,28,1,1
660 DATA 1,0,0,5,4,0,1,6,0,2,7,0,9,0,0,1
,7,0,1,0,0,13,4,0,2,2,0,1,0,0,1,2,0,11,4
,0,2,5,1
670 DATA 3,1,0,2,4,0,1,0,0,2,7,0,9,1,0,2
,7,0,1,1,0,13,4,0,1,2,0,3,1,0,11,4,0,2,5
,0,29,1,1
680 DATA 5,1,0,2,7,0,10,0,0,2,7,0,14,4,0
,4,1,0,11,4,0,1,5,0,30,1,1
690 DATA 4,5,0,1,7,0,12,5,0,2,7,0,13,4,0
,5,5,0,11,4,0,31,5,1
700 DATA 4,0,0,1,7,0,11,0,0,1,7,0,1,0,0,
13,4,0,7,0,0,10,4,1
710 DATA 4,0,0,1,7,0,10,0,0,2,7,0,1,0,0,
12,4,0,8,0,0,10,4,1
720 DATA 3,0,0,2,7,0,9,0,0,2,7,0,2,0,0,1
1,4,0,9,0,0,9,4,1
730 DATA 3,0,0,1,7,0,9,0,0,2,7,0,3,0,0,1
0,4,0,9,0,0,9,4,1
740 DATA 3,0,0,1,7,0,8,0,0,2,7,0,4,0,0,8
,4,0,1,5,0,10,0,0,8,4,1
750 DATA 3,0,0,1,7,0,6,0,0,3,7,0,5,0,0,1
,5,0,6,4,0,2,5,0,9,0,0,8,4,0,1,5,1
760 DATA 3,0,0,1,7,0,3,0,0,4,7,0,7,0,0,2
,5,0,4,4,0,4,5,0,8,0,0,7,4,0,2,5,1
770 DATA 3,0,0,6,7,0,10,0,0,9,5,0,8,0,0,
5,4,0,4,5,1
780 DATA 19,0,0,9,5,0,7,0,0,6,4,0,4,5,1
790 DATA 20,0,0,8,5,0,7,0,0,6,4,0,3,5,1
800 DATA 21,0,0,7,5,0,7,0,0,5,4,0,4,5,1
810 DATA 22,0,0,6,5,0,6,0,0,6,4,0,3,5,1
820 DATA 22,0,0,6,5,0,6,0,0,6,4,0,2,5,1
830 DATA 23,0,0,5,5,0,6,0,0,5,4,0,3,5,1
840 DATA 23,0,0,5,5,0,5,0,0,6,4,0,2,5,1
850 DATA 24,0,0,3,5,0,1,3,0,5,0,0,6,4,0,
1,5,1
860 DATA 23,0,0,4,2,0,1,3,0,5,0,0,5,4,0,
1,5,1
870 DATA 23,0,0,3,2,0,2,3,0,4,0,0,6,4,0,
1,5,1
880 DATA 22,0,0,3,2,0,3,3,0,4,0,0,5,4,0,
1,5,1
890 DATA 22,0,0,3,2,0,3,3,0,4,0,0,5,4,1
900 DATA 21,0,0,4,2,0,3,3,0,3,0,0,5,4,1
910 DATA 21,0,0,4,2,0,2,3,0,4,0,0,5,4,1
920 DATA 20,0,0,5,2,0,2,3,0,4,0,0,4,4,1
930 DATA 20,0,0,6,2,0,1,3,0,3,0,0,5,4,1
940 DATA 21,0,0,5,2,0,1,3,0,3,0,0,1,2,0,
3,4,1
950 DATA 21,0,0,4,2,0,2,3,0,2,0,0,3,2,0,
1,3,1
960 DATA 22,0,0,2,2,0,2,3,0,3,0,0,3,2,0,
1,3,1
970 DATA 22,0,0,2,2,0,1,3,0,4,0,0,2,2,0,
1,3,1
980 DATA 29,0,0,2,2,0,1,3,1
990 DATA 28,0,0,2,2,0,2,3,1
1000 DATA 28,0,0,2,2,0,2,3,1
1010 DATA 27,0,0,4,2,0,1,3,1
1020 DATA 27,0,0,4,2,0,2,3,1
1030 DATA 27,0,0,5,2,0,2,3,1
1040 DATA 27,0,0,5,2,0,2,3,1
1050 DATA 27,0,0,6,2,0,1,3,1
1060 DATA 27,0,0,6,2,1
1070 DATA 27,0,0,4,2,1
```

GRAPHICS MODE 11

Mode 11 is similar to mode 9 except instead of varying luminances and a SETCOLOR, mode 11 offers varying colors at a set luminance. To set the luminance type:

SETCOLOR 4,0, Luminance wanted.

A program demonstrating mode 11 is given below:

```
10 REM * GRAPHICS MODE 11 DEMO. *
20 GRAPHICS 11:SETCOLOR 4,0,8
30 COLOUR=INT(RND(0)*15)+1
40 COLOR COLOUR
60 PLOT X,Y:DRAWTO 79-X,Y:DRAWTO 79-X,19
1-Y:DRAWTO X,191-Y:DRAWTO X,Y
65 SOUND 0,X,10,15:SOUND 1,X+1,10,15:SOU
ND 2,Y,12,8
70 X=X+0.5:Y=Y+1
80 IF X<40 THEN 30
84 COLOR 0
90 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0
,0,0
100 GOTO 100
```

Chapter 3 : INPUT/OUTPUT ON THE ATARI

This chapter is going to deal with creating, reading and writing to files. Before you can use files, they must be OPENed. This is done by:

OPEN # (file number (1-5), MODE, 0, “DEVICE”

MODE	*MEANING*	*DEVICE*	*MEANING*
1	Input	“P:”	Printer
8	Output	“C:”	Cassette
12	Input + Output	“D:”	Disk
6	Disk directory	“S:”	Screen
9	End-of-file	“K:”	Keyboard
	Append	“R:”	RS232C
	Operaton	“E:”	Editor

An example: OPEN #3,6,9,“D:*.*” would open file number 3 the disk drive to read the directory. OPEN # 5,4,0,“C:” would open file number 5 to an input operation from the cassette.

Files can be read from/written to in many different ways. There is PUT, GET, INPUT #, PRINT #, and certain X10 comands.

PUT and GET

These commands PUT and GET single bytes. The format is PUT # file number (1–5), variable. An example would be PUT #2, DATABYTE or GET # 3,ZZ. A program demonstrating PUT and GET is listed below:

```
10 OPEN #1,4,0,"K:":POKE 752,1
20 OPEN #2,8,0,"P:":GRAPHICS 0
30 GET #1,BYTE
40 PUT #2,BYTE
50 PUT #6,BYTE
60 GOTO 30
```

Note, that in line 50 it is outputing to file number six even though there is no corresponding OPEN statement. This is because file number six is automatically opened when a GRAPHICS statement is encountered. The advantage of using the GET and PUT commands is that you are directly ordering single bytes, therefore being simpler and easier.

INPUT # and PRINT

These commands are used to transfer a string of characters. An example using these commands might be PRINT # 2; "ATARI";E$ or INPUT # 1,A$.

In the first example, "ATARI" will be written to file number 2, followed by the contents of E$. In the second example A$ will be filled with ATASCII characters until A$ is completely filled up or an EOL character is reached.

X10 5 and X10 9

These commands are virtually the same as INPUT # and PRINT # commands except that X10 9 will stop writing a string if an EOL character is reached, whereas PRINT # prints a string regardless of content. An example of X10 9 would be X10 9,#1,8,0,NAME$.

Chapter 4 :
THE DISPLAY LIST

The display list is a list of numbers inside ANTIC which tells the computer which graphics mode it is currently operating in. In BASIC, these numbers indicating the graphics mode are all the same. When you change the graphics mode by a GRAPHICS command, the display list numbers change. The length of the display list will vary with the different graphics modes, but in all the graphics modes there are 192 scan lines.

A scan line is a single line on the television screen. Certain graphics modes have different heights of the graphics rows. For example, a graphics row in mode 7 is much smaller than a graphics row in mode 3. To work out how high a graphics row is for a particular mode use this formula:

192 ÷ number of rows in full screen = height of graphics row in scan lines.
Example for graphics mode 5: no. of rows in full screen = 48 :
192 ÷ 48 = 4

So for graphics mode 5, each graphics row is exactly four scan lines high.

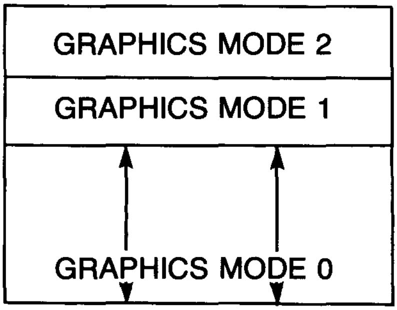
If we are to start changing the numbers in the display list, the graphics display will change. In this chapter we will discuss how to build your own display list, making your own graphics display. First of all, let me show you the display list for graphics mode 0.

```
All values in Hexadecimal
70 blank 8 lines
70 blank 8 lines     must be here to cover
70 blank 8 lines     overscan.
42 Display Mode 0 (ANTIC MODE 2)
20 }
7C }  Low, high bytes of start of screen memory at 7C20
02
02
02
02
02
02
02
02
02
02
02 }  Display ANTIC MODE 2 BASIC graphics MODE 0
02
02
02
02
02
02
02
02
02
02
41 }
E0 }  Jump to start display list which starts at 7BEO
7B }
Screen memory now follows.
```

As you can see, it looks quite complicated. All will be revealed soon. First, to find where the display list is. This is very simple, you just enter the command:

DISPLAY LIST = PEEK (560) + 256* PEEK (561) + 4

Before you go any further, you must work out what display you want. I am gong to give one example here:



```
10 GRAPHICS 0
20 DL=PEEK(560)+256*PEEK(561)+4
30 POKE DL-1,71:POKE DL+2,6
40 SETCOLOR 4,9,4
50 ? "the program that       DOES THIS IS
"
60 ? "LISTED BELOW........"
70 LIST
```

Program explanation:
Line 10: As out of the three graphics modes we are using, 1,2, and 0, Mode 0 uses the most memory. Thus, we must specify this graphics mode.

Line 20: This statement finds the start of the display list data for the particular graphics mode, and stores the number in the variable DL.
Line 30: If the first line of your required display is not the same as the graphics mode that uses the most memory, then you must POKE the value for the display list minus one with the correct numbers corresponding to the required graphics mode you want the first screen line to display. The numbers are listed below:

GRAPHICS MODE	0	1	2	3	4	5	6	7	8
NUMBER TO BE POKED	66	70	71	72	73	74	75	77	79

The second statement in line 30 puts the second screen line in mode 1. This is done simply by adding the number of screen lines to the variable DL and POKE this with this set of numbers corresponding to the required graphics mode for that particular screen line.

GRAPHICS MODE	0	1	2	3	4	5	6	7	8
NUMBER TO BE POKED	2	6	7	8	9	10	11	13	15

Line 40 gets rid of the black border.
Line 50 prints in orange and yellow, in graphics mode one and two.
Line 60 prints in graphics mode 0.
Line 70 lists the program in graphics mode 0.
The way you can really learn is to experiment, find out where the display list is, and POKE around. You're sure to find some special effects.

Chapter 5 :
THE ERROR SYSTEM

This is a short chapter on the error system inside your Atari home computer.

When the Atari computer comes across an error it will stop executing the program and will print an error message such as ERROR–6 AT LINE 20. You then look up in your manual what error number 6 is. You then find out that it means "Out of Data". You try your best then to amend your program so that the program works as it should.

Atari Basic provides you with a useful command called "TRAP". This command will force execution to be continued at the line specified when an error is encountered. An example: TRAP 520 will cause the computer to GOTO line 520 if an error occurs. If an error does occur then the error number can be found by PRINT PEEK (195). The error at which the line occurred can be found by PRINT PEEK (186) + 256 * PEEK (187).

A program that might prove useful to you is my error trapping program. Type it in and save it to disk or cassette. Now, whenever you are about to write a program, simply load it in. At the beginning of the program make sure that there is a TRAP 32000. Make sure that the line numbers do not exceed 31999 or the error trapping program will start being erased. When you have finished writing your program, simply save it to disk or cassette using either LIST "C:",0,31999 for cassette or LIST "D:FILE.EXT",0,31999 if you have disk. This prevents the error trapping routine being saved as well. When you want your program back in just use ENTER "C:" or ENTER "D:FILE.EXT" if you have disk. The error trapping routine now follows:

```
10 REM +* FIRST LINE MUST BE TRAP +*
20 REM *+ 32000 THEN YOUR PROGRAM *+
30 REM +* GOES HERE.               +*
32000 P=PEEK(195):ERL=PEEK(186)+256*PEEK
(187)
32010 TRAP 32000
32020 IF P=2 THEN ? "Out of memory ";
32030 IF P=3 THEN ? "Out of range ";
32040 IF P=4 THEN ? "Too many variables
";
32050 IF P=5 THEN ? "String length too l
arge ";
32060 IF P=6 THEN ? "Out of data ";
32070 IF P=7 THEN ? "Line number too big
 ";
32080 IF P=8 THEN ? "INPUT error ";
32090 IF P=9 THEN ? "Array/DIM error ";
32100 IF P=10 THEN ? "Stack overflow ";
32110 IF P=11 THEN ? "Arithmetic overflo
w ";
32120 IF P=12 THEN ? "Undefined statemen
t ";
32130 IF P=13 THEN ? "NEXT without FOR "
;
32140 IF P=14 THEN ? "Line too long erro
r ";
32150 IF P=15 THEN ? "No GOSUB or FOR ";
32160 IF P=16 THEN ? "RETURN without GOS
UB ";
32170 IF P=17 THEN ? "Syntax error ";
32180 IF P=18 THEN ? "Invalid string cha
racter ";
32190 IF P=136 THEN ? "End of file ";
32191 IF P=141 THEN GRAPHICS 0:? "Graphi
cs error ";
32200 IF P>18 AND P<>136 AND P<>141 THEN
 ? "INPUT/OUTPUT error ";
32210 IF ERL<>0 THEN ? "at line ";ERL
32220 IF ERL=0 THEN ?
32230 END
```

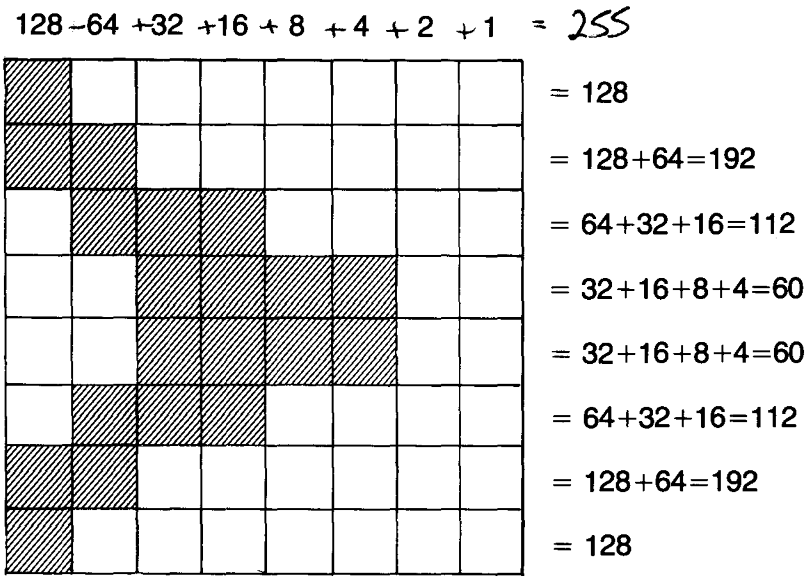
Note that if you use TRAP to a non-existant line number, an ERROR-12 will occur for any error. This is because when any error occurs, the computer checks if a TRAP has been set; if it has it will try to go to that line number. Thus, if it cannot, an error 12 will occur.

Chapter 6 : PLAYER-MISSILE GRAPHICS

I, personally, think that this capability is the best thing that is offered by the Atari computers. It makes Atari shine out from the rest. With a little effort, you will have players and missiles performing at the snap of your fingers.

There are four players and four missiles, each with their own color register, size register and horizontal position register. You can have players going under certain playfields and over other playfields. Player missile graphics reside in RAM and so take up memory. Having single line resolution (one line of a player is one scan line) takes up exactly 2K of memory. Having double line resolution (each line of a player is two scan lines) takes up 1K of memory.

A player is exactly 8 bits wide by either 128 (double) or 256 (single line resolution). To create your player, you just plot the squares on a grid 8 by however tall you want your player. Then put the numbers 128, 64, 32, 16, 8, 4, 2 and 1 (the binary headings) above the grid. Then add up the binary headings if the square is filled in. An example is:



A program that will put this player on the screen and so that you can move it about on the screen is listed below. Pressing the trigger button will change the size of the player. Type it in, see how it works and perhaps change it to suit you. A detailed explanation of how the program works is also given. Study it, and experiment, try to see how it works.

```
10 REM ** PLAYER-MISSILE GRAPHICS **
20 REM ** EXAMPLE BY PAUL BUNN.   **
30 GRAPHICS 0:SETCOLOR 2,0,0:RAMTOP=PEEK
(106)-8
40 RAM=RAMTOP*256
50 FOR X=512 TO 640:POKE X+RAM,0:NEXT X
60 FOR P=60 TO 67:READ X:POKE 512+P+RAM,
X:NEXT P
70 DATA 128,192,112,60,60,112,192,128
80 POKE 53248,128:POKE 53277,3:POKE 559,
46:POKE 54279,RAMTOP
90 XPOS=128:YPOS=60:SIZE=0
100 S=STICK(0)
110 XPOS=XPOS+(S=7)-(S=11)
120 POKE 53248,XPOS:POKE 704,XPOS
130 IF STRIG(0)=0 THEN FOR P=255 TO 0 ST
EP -20:SOUND 0,P,12,12:NEXT P:SIZE=SIZE+
1:SOUND 0,0,0,0
140 IF SIZE>3 THEN SIZE=0
150 POKE 53256,SIZE
160 IF S=14 THEN FOR P=0 TO 8:POKE RAM+5
11+YPOS+P,PEEK(RAM+512+YPOS+P):NEXT P:YP
OS=YPOS-1
170 IF S=13 THEN FOR P=8 TO 0 STEP -1:PO
KE RAM+512+YPOS+P,PEEK(RAM+511+YPOS+P):N
EXT P:YPOS=YPOS+1
180 GOTO 100
```

Program Explanation:
Line 30: Clears the screen, sets the background to black so that you can see the player more clearly. Location 106 holds the amount of memory pages you have, this is stored in a variable RAMTOP. 8 pages are stepped back for the player/missile area.
Line 40: RAM is to hold the actual RAM location at which the player/missile arrangement starts.
Line 50: This clears out the data currently held in PLAYER 0's place.
Line 60: This POKE's in the player.
Line 70: The data is decimal form for the binary layout for the player.
Line 80: POKE location 53248 with the horizontal value for player 0, in this case, 128 for the middle of the screen. Location 53277 is POKEd with a 3. This enables players and missiles. Location 559 is POKEd with a 46. This specifies double line resolution. Location 54279 is POKEd with variable RAMTOP. This tells the computer where the player/missile arrangement starts.
Line 90: Sets x-position and y-position variables to centre of screen. Also sets size to small.
Line 100: Reads value of joystick.
Line 110: Increases or decreases the x-position variable according to whether the joystick is pushed to the right or left.
Line 120: POKE the horizontal position register for player 0 with the x-position variable. Also POKE the color of player 0 with the x-position variable as well.
Line 130: Check if the red button is pressed down, if it is then change size of player.
Line 140: Check if size variable is greater than three.

When size = 0 size of player = NORMAL
When size = 1 size of player = DOUBLE
When size = 2 size of player = NORMAL
When size = 3 size of player = QUADRUPLE

Line 150: POKE the size of player 0 register with the variable size.

Line 160: Check if joystick is moved up; if it is then move player up.
Line 170: Check if joystick is moved down; if it is then move player down.
Line 180: Continue program loop.

Here is a list of all the locations that affect players and missiles:

LOCATION	*NAME*	*COMMENTS*
559	SDMCTL	POKE a 46 here for double, 62 for single line resolution.
623	GPRIOR	This location determines priorities for players and playfied. POKE a 1 : Players have priority over playfields 4 : Plafields have priority over players 2 : Players 1 and 0 then playfield Players 2 + 3 8 : Playfield 0 and 1 then players, Playfield 2 + 3
704	PCOLRO	Color of player-missile 0
705	PCOLR1	Color of player-missile 1
706	PCOLR2	Color of player-missile 2
707	PCOLR3	Color of player-missile 3
53277	GRACTL	Graphic control 1: A 3 here will enable player-missile graphics whereas a 0 will disenable them.
53278	HITCLR	POKE any value here to clear all collision detection registers.
53248	M0PF	Missile 0 to playfield collisions.
53256	M0PL	Missile 0 to player collisions
53249	M1PF	Missile 1 to playfield collisions

LOCATION	*NAME*	*COMMENTS*
53257	M1PL	Missile 1 to player collisions
53250	M2PF	Missile 2 to playfield collisions
53258	M2PL	Missile 2 to player collisions
53251	M3PF	Missile 3 to playfield collisions
53259	M3PL	Missile 3 to player collisions
53252	POPF	Player 0 to playfield collisions
53260	POPL	Player 0 to player collisions
53253	P1PF	Player 1 to playfield collisions
53261	P1PL	Player 1 to player collisions
53254	P2PF	Player 2 to playfield collisions
53262	P2PL	Player 2 to player collisions
53255	P3PF	Player 3 to playfield collisions
53263	P3PL	Player 3 to player collisions
53252	HP0SM0	Horizontal position of missile 0
53253	HP0SM1	Horizontal position of missile 1
53254	HP0SM2	Horizontal position of missile 2
53255	HP0SM3	Horizontal position of missile 3
53248	HP0SP0	Horizontal position of player 0
53249	HP0SP1	Horizontal position of player 1
53250	HP0SP2	Horizontal positoin of player 2
53251	HP0SP3	Horizontal positon of player 3
54279	PMBASE	This is the highest byte of where the player missile area starts.
53260	SIZEM	Size of all missiles
53256	SIZEP0	Size of player 0
53257	SIZEP1	Size of player 1
53258	SIZEP2	Size of player 2
53259	SIZEP3	Size of player 3
		POKE an X here for size registers: 0 = NORMAL 1 = DOUBLE WIDTH 2 = NORMAL 3 = QUADRUPLE WIDTH

HORIZONTAL MOVEMENT OF PLAYERS AND MISSILES

This kind of movement is the most simplest for player-missile graphics. There are horizontal position registers for all four missiles and players. All you need to do is to POKE the corresponding horizontal position register with the horizontal position. POKEing a low number will move that missile or player to the left at the speed of light! POKEing a high number will move it to the right. Due to overscan, players and missiles will be off the screen between numbers less than 50 or greater than 200. These parameters may vary a great deal between different televisions or monitors. So POKE a 50 for the far left hand side of the screen and a 200 for the far right hand side.

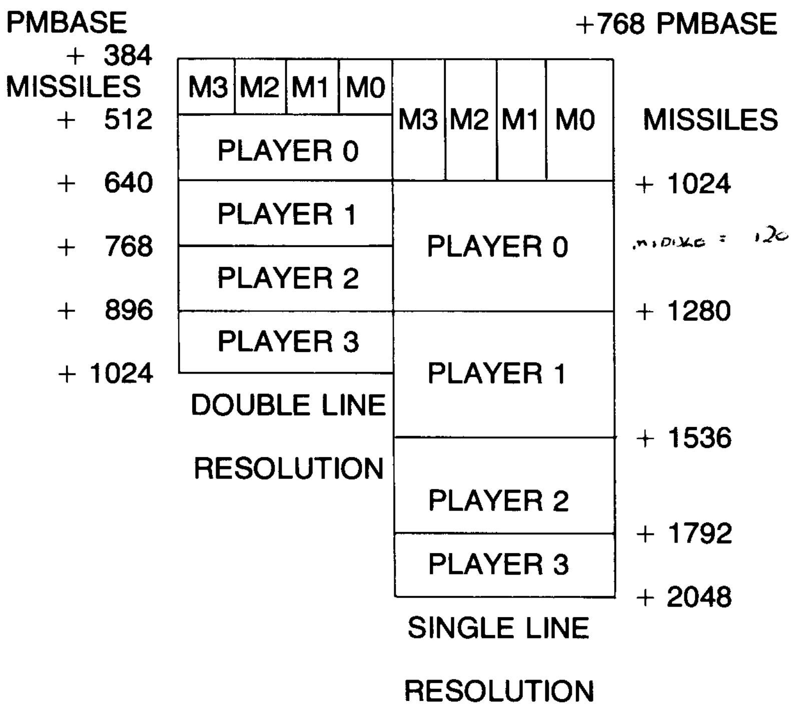
VERTICAL MOTION WITH PLAYERS AND MISSILES

This kind of movement is a little more difficult, and quite slow in BASIC. I consider players and missiles to be a ‘band’ of memory running from the very top of the television screen to the very bottom. This ‘band’ will be on the X-axis corresponding to its horizontal position register. It will always be 8-bits wide (except missiles which are two bits wide). The width of each bit will be affected by its corresponding size register.

POKEing anywhere in this band will immediately display the corresponding byte in binary form. So, to actually move a player down or up you simply have to move it up or down in memory.

A FIFTH MULTIPLE COLOR PLAYER

A multiple color player can be made by joining the four missiles, which all have their independent color register. All you have to do is to build your player in the four missiles area. Then put the required color for the two right most bits in the color register for missile 0, the next 2 bits in the color register for missile 1 and so on. Vertical motion is the same, but with horizontal motion all the horizontal positon registers must be in order (M3, M2, M1 and then M0), must be sequential.



860 TO 867

MIDDLE = 120 TO 127

If, for example, using double line resolution the hi-byte of PMBASE is 54 then in single line resolution the player missile area will start at 14208 in RAM (256 multiplied by 54) + 384.

COLLISION DETECTION

Before detecting a collision, first POKE HITCLR with any value. This clears all the collision registers. Then PEEK at the required collision detection register. Anything other than 0 means that a collision has taken place.

Chapter 7 : REDEFINING THE CHARACTER SET

This is another useful application which the Atari computer offers. The data for the original character set is stored in ROM in binary form. The table starts at 57344 and is 1024 bytes long. You can only PEEK into the table and not POKE into it. Well, how do you change it then? Well, in RAM is a location called CHBAS which stands for character base register. You can change CHBAS to what you want.

CHBAS is a hi-byte or MSB to the pointer of the character set table. When the value is 224 (the default) value) it points to 57344. What you have to do is to copy the table from ROM into RAM, so then you can change and alter the RAM data as much as you like.

A program that does this and redefines the exclamation mark character is listed below. A detailed explanation is also given so that you can see how it works.

```
10 REM ** THIS PROGRAM REDEFINS THE **
20 REM ** EXCLAMATION MARK CHARACTER**
30 RAMTOP=PEEK(106)-8:POKE 756,RAMTOP
40 FOR X=0 TO 1023:POKE RAMTOP*256+X,PEE
K(57344+X):NEXT X
50 LOC=RAMTOP*256+8
60 FOR X=0 TO 7
70 READ BYTE
80 POKE LOC+X,BYTE:NEXT X
90 DATA 204,204,51,51,204,204,51,51
```

Line 30: Find top of memory and step back 8 pages. POKE 756 with this.
Line 40: Put the ROM character set into RAM. When you first run the program you'll see this happening.
Line 50: Finds the start of the 8 byte data for the exclamation mark character.
Line 60-90: POKE in data for the character to replace the exclamation character.

When 'READY' appears, press shift and '1' together and see the new character.
To redefine the character of your choice you have to look up the character in the internal character chart (in the Atari reference manual) multiply that number by eight and add it to where your character set starts in RAM. For example, in the program:

```
LOC = RAMTOP *256+8
```

RAMTOP*256 finds the beginning of the character set. Add 8 because the "!" character is the first character (NUMBER 1). Note that number 0 is a space.

Chapter 8 : ATARI'S YELLOW CONSOLE KEYS

This is going to be a very short chapter on how to read the yellow Console Keys. Any combination of START, OPTION and SELECT can be found by a single PEEK location. The location to PEEK at is 53279. The table below shows which keys are pressed at a certain number between 0 and 7.

PEEK VALUE / KEYS PRESSED	0	1	2	3	4	5	6	7
START	●		●		●		●	
SELECT	●	●			●	●		
OPTION	●	●	●	●				

A dot ● means that the corresponding key has been pressed. An example, when PEEK (53279) gives 2 means that both START and OPTION keys have been held down. A program which puts this information to use is given below. POKE 53279,0 makes the actual computer 'click'.

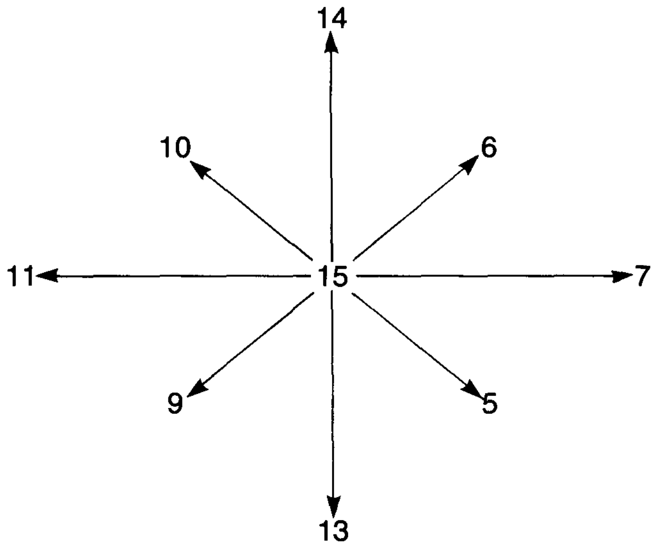
```
10 GRAPHICS 0:DIM CONSOL$(3):POKE 752,1
20 POKE 53279,0
30 KEY=PEEK(53279)
40 ON KEY GOTO 80,50,60,70,80,90
50 IF KEY=0 THEN 70
60 POSITION 0,0:? "NO KEYS ARE PRESSED":
CONSOL$="000":GOTO 150
70 CONSOL$="111":GOTO 140
80 CONSOL$="110":GOTO 140
90 CONSOL$="101":GOTO 140
100 CONSOL$="100":GOTO 140
110 CONSOL$="011":GOTO 140
120 CONSOL$="010":GOTO 140
130 CONSOL$="001":GOTO 140
140 POSITION 0,0:? "                   "
150 POSITION 2,5:IF CONSOL$(1,1)="1" THE
N ? "OPTION":GOTO 170
160 ? "      "
170 POSITION 2,10:IF CONSOL$(2,2)="1" TH
EN ? "SELECT":GOTO 190
180 ? "      "
190 POSITION 2,15:IF CONSOL$(3,3)="1" TH
EN ? "START":GOTO 20
200 ? "     "
210 GOTO 20
```

Chapter 9 :
JOYSTICKS AND PADDLES

On the Atari computer, to read the joysticks or paddles is as simple as ABC. There are two commands, STICK and PADDLE which give you access to the games' controllers. You do not have to PEEK or POKE like you have to on some computers; a chip called POKEY does it all for you.

The STICK command

The STICK command is followed by a number in brackets ranging from 0–3. This number refers to the joystick to be read. Number 0 is joystick one and number 3 is joystick 4. The value that the STICK command gives relates to the way that joystick is pointing.



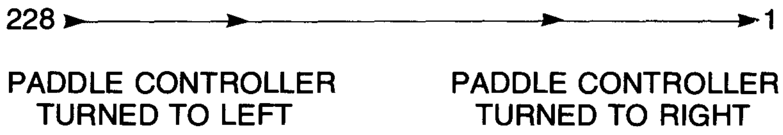
How to check if the read button on the Joystick Controller is pressed.

The command that checks the red button on the joystick is STRIG. There is also a number following this command in brackets. For example: STRIG(2) would refer to the red button on joystick number 3. Try putting a joystick in socket one and typing PRINT STRIG(0). If you had the button pressed down when you entered 'RETURN' the computer will PRINT a zero. Otherwise the computer will PRINT a one.

How to read the Paddle Controllers.

Up to eight paddle controllers can be attached to the Atari computer through only four ports. How? Well, each pair of paddles has only one socket plug. The paddles are numbered from 0–7 from left to right. An example: PADDLE(2) will give you the status of paddle controller number three. The status can range from 1 to 228. The diagram demonstrates this:

PADDLE command gives:


228
1
PADDLE CONTROLLER TURNED TO LEFT
PADDLE CONTROLLER TURNED TO RIGHT

Checking the red button on the Paddles

To check the red button on the paddle controllers is very similar to checking the red button on the joystick controllers, except the command to use is PTRIG. This command works in the same way as the STRIG function, except the number in brackets ranges from zero to seven indicating the paddle number.

Chapter 10 : MAKING PROGRAMS MORE EFFICIENT

Writing Programs

Most books on general programming suggest you start by drawing up a 'flowchart'–a pretty combination of circles, diamonds and slanting rectangles–which sets out the path and operations the computer will follow to execute the program. In theory, this is fine but in practice, especially when using a computer like the Atari which can quickly pinpoint errors, and will not 'self-erase' if you've made a programming mistake, the time and trouble involved is probably not worth it.

It is, however, essential to know exactly what you want the computer to do before you start creating a new program, even if you're not quite sure how you are going to get the Atari to carry out the task.

Sometimes a rough sort of flowchart–just the main steps the Atari wil take, linked by lines and loops–will help to clarify your thinking. This is also a good way of spotting potential problems (such as infinate loop) or not specifying the computer's task exactly.

We generally work out the 'core' of a program on paper, then enter it into the Atari, starting at line 100 to leave room before this line to assign variables, arrays and the like. If you're working on a fairly simple game, or are adapting one from a magazine or book, it is just as well to work directly on the Atari, but you could keep a notebook handy to record things like the fact that, for example, N$ is for the player's name.

If you have added memory, or memory to spare, don't let it s presence lead you into sloppy programming habits. It is very easy to set up a long and clumsy set of IF statements which really should be replaced by an IF then a GOSUB. When you have memory to spare it seems too much trouble to bother cleaning up your programs.

Making Games Programs more Interesting

The best thing you can do for a longer program is to add the element of surprise. If you can include situations which do not occur every time a game is played, you'll ensure the game will remain interesting for a much longer time than would be the case if every situation was triggered each time a game was run.

Another idea is to offer a degree of difficulty feature. Make sure that this feature really does increase the difficulty of the game, and ensure that–even at the highest level of play–the final score is reasonably attainable.

You can add even more interest to games by awarding points, scores, ratings or whatever, that are genuinely related to the speed and skill the player demonstrated. A further twist is to award a 'rank'. Atari's Star Raiders' cartridge does this by giving you a rating from 'garbage scow captain' to 'commander'. I bet you that this ranking system is what kept you playing Star Raiders 24 hours a day!

Making programs run faster and how to save Memory

1. Try to re-structure a BASIC program so that it is as efficient as possible. This will lower execution time and save memory.
2. Put all frequently used subroutines and FOR + NEXT loops at the beginning of the program. As BASIC tries to reference a sub-routine by it's line number, any subroutine at the beginning of the program will take less time to reach. At the very first line make this a GØTØ statement pointing to the main program loop.
3. If a very short subroutine is used, delete the subroutine and put the statements from the subroutine into the place where the GOSUB command was issued.

An example:

```
10 FØR X = 1 TØ 10
20 GØSUB 50
30 NEXT X
40 END
50 PRINT "HELLØ" : RETURN
```

would be changed to:

```
10 FØR X = 1 TØ 10
20 PRINT "HELLØ"
30 NEXT X
40 END
```

Doing this, not only saves memory, but decreased execution time (speeds the program up).

4. If the screen data is not required, by turning off the screen, the program will run 30% faster. Turn off the screen by the command: POKE 559,0. When you want the screen back on just enter the GRAPHICS command.
5. If you understand machine code, use the USR function. Assembly language is much faster than BASIC.
6. Remove all the REM statements. REM statements just take up space in the program.
7. If a number is referenced many times, then 6 bytes can be saved if a variable is set to that number, and the variable is referenced instead.
8. At the start of a program, variables can be set up by a READ/DATA statement. This will save memory because data in the DATA statements will be stored as ATASC11 one byte date.
9. Use the colon character to have more than one statement on a line.
10. Use a POKE instead of SETCOLOR statements.
11. Use a POKE instead of POSITION statements.
12. Use a POKE instead of STICK, PADDLE, PTRIG, STRIG, statements. Where to POKE is given in the memory locations section of this book.
13. An idea may be to chain programs. An example may be that the first program sets all variables, strings and the like, and the main program actually makes the program go.

Chapter 11 : THE SOUND COMMAND

The SOUND command is something offered by the POKEY chip. What I like about Atari sound is that the sound is playable through the television speaker and not through the tiniest loudspeaker you've ever seen in the computer. Another thing I like about it is that you have four completely independent sound registers.

As you probably know, the SOUND command has four parameters:
SOUND channel, tone, distortion, volume.

Channel:	This is a number referring to the four channels offered. The number ranges from zero to three.
Tone:	A number ranging 0–255. The higher the number the lower the note.
Distortion:	This number is an even number from 0–14, although you can have a number from 0–255. A number of 10 means 'pure tone'. Other numbers give special effects.
Volume:	A number between 0–15, the higher the number, the louder the note.

You can create a good effect by all the channels playing similar notes. Try typing this:

```
SOUND 0,200,10,15
SOUND 1,201,10,15
SOUND 2,202,10,15
SOUND 3,203,10,15
```

This gives quite an amusing effect.

Another interesting effect can be created by rapidly lowering the volume of a note. The program following does this:

```
10 REM ++** SOUND DEMO ++**
20 FOR TONE=255 TO 0 STEP -20
30 FOR VOLUME=15 TO 0 STEP -2
40 SOUND 0,TONE,10,VOLUME
50 NEXT VOLUME
60 NEXT TONE
70 END
```

To turn off all sounds, use END, or to turn off individual sounds use SOUND channel, 0,0,0.

LOCATION 540

This location in memory is a count-down timer. Every 1/60 of a second, it's value is decreased by one (decremented). Usually it's value is 0, so nothing is changed. Location 540 can be used to time a sound for up to nearly 4 1/2 seconds. The following short program plays each note for two seconds.

```
10 REM ++** SOUND DEMO 2 ++**
20 FOR TONE=255 TO 0 STEP -20
30 POKE 540,120:REM 120=2 SECONDS*60
40 SOUND 0,TONE,10,15
50 IF PEEK(540)<>0 THEN 50
60 NEXT TONE
70 END
```

Chapter 12 : MEMORY LOCATIONS

Before I go any further, you must firstly fully understand the esence of PEEK and POKE. Here are examples:

POKE location, BYTE where BYTE is a number between 0–255 LET A = PEEK (BYTE) or ? PEEK (2036)
POKE is used to change the contents of any location in RAM.
PEEK is used to return the contents of a location.

Another thing you must understand is a two byte address. Say that there is a two byte at locations 14 and 15. To find their address in the example would be:

```
ADDRESS = PEEK (14) + 256 * PEEK(15)
```

NAME	*DECIMAL ADDRESS*	*HEX. ADDRESS*	*DESCRIPTION*
APPHI	14,15	D.E	This is a two-byte adress pointing to the highest location used by BASIC.
POKMSK	16	10	Interrupt request enable
RTCLOK	18,19,20	12,13,14	TV frame counter.
SOUNDR	65	41	NOISY I/O FLAG
ATTRMOD	77	40	Attract mode −128 = YES
LMARGIN	82	52	Left screen margin
RMARGIN	83	53	Right screen margin
ROWCRS	84	54	Cursor row in graphics window
COLCRS	85,86	55,56	Cursor column in graphics window
DATCURS	93	5D	graphics mode 0, data under the cursor
NEWROW	96	60	Row to which DRAWTO will go to
RAMTOP	106	6A	Number of pages of RAM
LOMEM	128,129	80,81	BASIC low memory pointer
MEMTOP	144,145	90,91	BASIC memory top pointer
STOPLN	186,187	BA,BB	STOP/TRAP line number
ERRSAV	195	C3	ERROR NUMBER
PTABW	201	C9	PRINT TAB WIDTH– DEFAULT 10
FRO	212,213	D4,D5	Value to be returned after USR.
RADFLG	251	FB	RAD/DEG flag 0 = RAD 6 = DEG

SDMCTL	559	22F	OS DIRECT MEMORY ACCESS CONTROL
SDLSTL	560,561	230,231	Pointer to beginning of display list
SSKCTL	562	232	OS SERIAL PORT CONTROL
LPENH	564	234	LIGHT PEN HORIZONTAL VALUE
LPENV	565	235	LIGHT PEN VERTICAL VALUE
SYRES	580	244	When this is anything but a 0, then when system reset is pressed, a cold start will execute.
GPRIOR	623	26F	OS PRIORITY SELECT
PADDLO-7	624-277	270-277	PADDLE controllers 0 through 7
STICKO-3	632-635	278-27B	JOYSTICK controllers 0 through 3
STRIGO-3	644-647	284-287	JOYSTICK TRIGGERS 0 through 3
TXTROW	656	290	CURSOR row in text window
TXTCOL	657,658	291,292	CURSOR column in text window
TXTWND	703	2BF	TEXT WINDOW 24=OFF 4=ON
PCOLR0	704	2C0	Color of player-missile 0
PCOLR1	705	2C1	Color of player-missile 1
PCOLR2	706	2C2	Color of player-missile

			2
PCOLR3	707	2C3	Color of player-missile 3
COLOR0	708	2C4	Color of playfield register 0
COLOR1	709	2C5	Color of playfield register 1
COLOR2	710	2C6	Color of playfield register 2
COLOR3	711	2C7	Color of playfield register 4
MEMTOP	741,742	2E5,2E6	OS MEMORY TOP POINTER
MEMCO	743,744	2E7,2EB	OS MEMORY LOW POINTER
CRSINH	752	2F0	A 0 here for the cursor to be displayed a 1 to turn it off.
CHACT	755	2F3	A 4 here for text to be displayed upside down
CHBAS	756	2F4	A 224 here for upper-case characters, or a 226 for lower case. You can also POKE a value here to point to your own character set in RAM
ATACHR	763	2FB	Last ATASCII character
CH	764	2FC	Last key hit in internal code
FILDAT	765	2FD	Color that FILL operation is to use
CONSOL	53279	D01F	Location to be used when reading from the yellow consol keys

STIMER	53769	D209	Start timer
RANDOM	53770	D20A	Random number generator producing a number between 0–255 every 1/60th second
PACTL	54018	D302	A 52 here to turn the program recorder on, a 60 to turn it off
PENH	58284	D40C	The horizontal value of the light pen
PENV	58285	D40D	The vertical value of the light pen

Vectors

HEX.	*EC.*	
OA	10	DOS Vector
OC	12	Warm-start vector
12,13,14	18,19,20	3 bytes of built in clock
58	88	Pointer to the beginning of screen RAM
200	512	Display list interrupt vector
E453	58451	Disk interface
E45C	58460	This routine sets vectors
E45F	58463	This vector is executed at vertical blank
E462	58466	This vector is executed after vertical blank
E465	58469	Routine initialises serial input/output
E471	58481	Goes into the blackboard mode
E474	58484	Warm start
E477	58487	Cold start (power-up procedure)
E47A	58490	This routine reads a block of 128 data bytes into CASBUF (data starts at 400 in hex.)
E470	58493	Open cassette for input vector

Chapter 13:
GAMES PROGRAMS

The following programs are intentionally designed purely for the purpose of fun, although I hope you'll enter the programs as we have listed them here, and enjoy playing them. The real value of this section though, lies in what you do with (a) the programs to adapt them to make them your own; and (b) the ideas you get from them to use in new programs of your own.

Unfortunately, the ATARI printer does not print graphics characters, so I have had to ink in the appropriate graphics characters. Notice that the *( )* symbol is the clear screen symbol.

BEETLE JUICE

This is one of my favourite programs, about a poor red beetle. Instructions are listed below. Make sure that you SAVE the program first before you attempt to RUN it at all, because an error in the program, especially in the DATA statements could cause the computer to crash (because it uses assembly language in the form of USR calls).

```
The instructions for beetle juice:-
       B E E T L E     J U I C E
       *-*-*-*-*-*     *-*-*-*-*

 You are a small red beetle whose only
desire is to get to the other side of
the road.This is no easy task because
unfortunately for your beetle,there
are some maniac drivers around trying
to put Beetle Juice back on the menu
again!


 You may move up or down in your
desperate plot to get to the other
side.The ultimate goal is to get 10
beetles across.

 Scoring is for the time taken to get
the beetles safely home.
A score of less than 3000 is poor
about 5000 is avarage and
greater than 7500 is excellent.

HIT ANY KEY TO PLAY BEETLE JUICE!
```

```
10 REM ** B E E T L E   J U I C E **
          ************   **********
                 BY PAUL BUNN
20 GRAPHICS 0:POKE 752,1:DIM A$(1)
30 SETCOLOR 2,3,4:SETCOLOR 4,3,4:? "Do y
ou desire instructions ";
40 TRAP 20:INPUT A$:IF A$="Y" THEN 730
50 IF A$<>"N" THEN ? :? " PARDON !":? :G
OTO 30
60 TRAP 40000:GOTO 80
70 CLR :GOTO 20
80 GRAPHICS 1:POKE 756,226:SETCOLOR 0,0,
0:? #6;"                   ",,
90 FOR C=1 TO 5:? #6,,"---------------
---":NEXT C:? #6,,"                   "
100 SETCOLOR 2,4,4
110 GOSUB 480:POKE 752,1
120 RESTORE 670:FOR C=0 TO 98:READ D:POK
E 1536+C,D:NEXT C
130 POKE 53278,2:POKE 540,255
140 C=USR(1536):E=STICK(0):IF E=14 THEN
SOUND 0,70,10,15:C=USR(1595):SOUND 0,0,0
,0
150 IF E=13 AND PEEK(206)<216 THEN SOUND
 0,200,10,15:C=USR(1615):SOUND 0,0,0,0
160 IF PEEK(206)=141 THEN GOTO 190
170 IF PEEK(53263)<>0 THEN 280
180 GOTO 140
190 REM ** YOU WIN **
200 G=G+10*PEEK(540)
210 FOR D=252 TO 4 STEP -4:SOUND 0,D,10,
15:SOUND 1,D+1,10,15:SOUND 2,D+2,10,15:S
OUND 3,D+3,10,15
220 NEXT D
230 FOR C=0 TO 3:SOUND C,0,0,0:NEXT C
240 F=F+1:? CHR$(125);"     BEETLES HOME
 SAFELY : ";F
250 IF F=10 THEN 400
260 X=USR(1615):IF PEEK(206)<216 THEN 26
0
```

```
270 GOSUB 580:GOTO 130
280 ? "BEETLE JUICE IS ON THE MENU AGAIN
!"
290 FOR D=1 TO 45:C=USR(1536):POKE 707,P
EEK(53770)
300 POKE INT(RND(0)*6)+PEEK(206)+256*PEE
K(207),PEEK(53770)
310 SOUND 0,RND(0)*80,80,15:NEXT D
320 POKE 707,0:SOUND 0,0,0,0
330 ? CHR$(125);"   Hit any key for anot
her go."
340 ? "  SCORE=";G:G=0
350 POKE 764,255:GOSUB 580
360 IF PEEK(764)=255 THEN 360
370 POKE 707,54:C=USR(1595)
380 C=USR(1615):IF PEEK(206)<216 THEN 38
0
390 F=0:GOTO 130
400 GRAPHICS 2:SETCOLOR 2,0,0:? #6;" con
gratulations!!"
410 ? #6,,,,"YOU HAVE MANAGED TO"
420 ? #6,,"GET 10 BEETLES HOME        SA
FELY"
430 ? #6,,"well done indeed!!!"
440 POKE 53248,0:POKE 53249,0:POKE 53250
,0:POKE 53251,0
450 ? #6;"score=";G
460 FOR C=0 TO 255:SOUND 0,C,12,15:NEXT
C
470 SOUND 0,0,0,0:END
480 A=PEEK(106)-8:POKE 54279,A:B=256*A:F
OR C=B+512 TO B+1024:POKE C,0:NEXT C
490 RESTORE 660:FOR C=B+531 TO B+538:REA
D D:POKE C,D:NEXT C
500 RESTORE 600:FOR C=B+670 TO B+677:REA
D D:POKE C,D:NEXT C
510 RESTORE 620:FOR C=B+810 TO B+817:REA
D D:POKE C,D:NEXT C
520 RESTORE 630:FOR C=B+566 TO B+573:REA
D D:POKE C,D:NEXT C
```

```
530 RESTORE 640:FOR C=B+707 TO B+714:REA
D D:POKE C,D:NEXT C
540 RESTORE 630:FOR C=B+847 TO B+854:REA
D D:POKE C,D:NEXT C
550 RESTORE 610:FOR C=B+986 TO B+992:REA
D D:POKE C,D:NEXT C
560 C=INT((B+986)/256):D=B+986-256*C:POK
E 206,D:POKE 207,C
570 POKE 559,46:POKE 53277,3:POKE 623,1:
POKE 704,12*16+8:POKE 705,7*16+8:POKE 70
6,23:POKE 707,3*16+6
580 E=53770:POKE 203,PEEK(E):POKE 204,PE
EK(E):POKE 205,PEEK(E):POKE 53251,180
590 RETURN
600 DATA 24,113,254,191,191,254,113,24
610 DATA 24,90,60,126,126,60,66
620 DATA 238,68,238,123,123,238,68,238
630 DATA 231,66,218,127,127,218,66,231
640 DATA 119,34,119,222,222,119,34,119
650 DATA 231,66,91,254,254,91,66,231
660 DATA 24,142,127,253,253,127,142,24
670 DATA 104,230,203,169,200,197,203,176
,4,169,50,133,203,198,204,234,234,169,50
,197
680 DATA 204,144,4,169,200,133,204,230,2
05,230
690 DATA 205,234,234,169,200,197,205,176
,4,169
700 DATA 50,133,205,165,203,141,0,208,16
5,204,141,1,208,165,205,141,2,208,96
710 DATA 104,198,206,208,2,198,207,160,0
,185,89,6,145,206,200,192,9,208,246,96,1
04,230,206,208
720 DATA 2,230,207,76,66,6,0,24,90,60,12
6,126,60,66,0,0
730 ? "} B E E T L E   J U I C E"
740 ? " *-*-*-*-*-*    *-*-*-*-*"
750 ? :? " You are a small red beetle wh
ose only";
760 ? "desire is to get to the other sid
```

```
e of the road.This is no easy task becau
se"
770 ? "unfortunately for your beetle,the
re"
780 ? "are some maniac drivers around tr
ying"
790 ? "to put Beetle Juice back on the m
enu"
800 ? "again!":? :? " You may move up or
 down in your"
810 ? "desperate plot to get to the othe
r"
820 ? "side.The ultimate goal is to get
10"
830 ? "beetles across."
840 ? :? " Scoring is for the time taken
 to get"
850 ? "the beetles safely home."
860 ? "A score of less than 3000 is poor
"
870 ? "about 5000 is avarage and"
880 ? "greater than 7500 is excellent."
890 ? :? "HIT ANY KEY TO PLAY BEETLE JUI
CE!"
900 POKE 764,255
910 IF PEEK(764)=255 THEN 910
920 GOTO 80
930 END
```

SMASHOUT!

In this arcade style game, you have to pulverise as many bricks as you can using your paddle at the bottom of the screen. The paddle is controlled by the joystick by moving left and right. Use your paddle to keep the ball in play. You score points for destroying bricks, but look out, if the ball hits the top wall, your bat is reduced to half size!

When you type in the program, SAVE it before you attempt to RUN it because the computer will crash if you have typed in the data wrongly.

After RUNning the program, be patient for a while as the computer redefines the character set, POKEs in machine code and sets up player/missile graphics.

```
10 REM * SMASHOUT - WRITTEN BY      *
20 REM * PAUL BUNN SEPTEMBER 1982   *
30 ? CHR$(125);"PLEASE WAIT WHILE I REDE
FINE CHARS":BLS=3:K=142/19:MODE=1
40 ? "AND SET UP PLAYER/MISSILE GRAPHICS
"
50 ? ".........THANK YOU.........":? :?
:? "PLEASE HIT THE 'START' KEY TO BEGIN"
60 FOR P=1536 TO 1536+140:READ X:POKE P,
X:NEXT P
70 DATA 104,173,120,2,201,7,208,4,230,20
3
80 DATA 230,203,201,11,208,5,198,203,198
,203
90 DATA 24,165,203,141,0,208,201,56,176,
4
100 DATA 169,56,133,203,24,165,203,201,1
84,176
110 DATA 2,133,203,165,206,201,1,208,2,1
98
120 DATA 208,201,2,208,2,230,208,165,208
,141
130 DATA 1,208,201,197,144,4,169,1,133,2
06
140 DATA 165,208,201,57,176,4,169,2,133,
206
150 DATA 165,207,201,1,208,22,230,204,16
5,204
160 DATA 201,0,208,2,230,205,160,0,185,1
37
170 DATA 6,145,204,200,192,4,208,246,165
,207
180 DATA 201,2,208,22,198,204,165,204,20
1,0
190 DATA 208,2,198,205,160,0,185,137,6,1
45
200 DATA 204,200,192,4,208,246,96,0,128,
128,0
210 IF PEEK(53279)<>6 THEN POKE 53279,0:
POKE 53279,8:GOTO 210
220 DIM G$(9):G$="GAME OVER"
```

```
230 GRAPHICS 1+16:RAM=PEEK(106)-8:FOR X=
0 TO 1023:POKE RAM*256+X,PEEK(57344+X):N
EXT X
240 POKE 756,RAM:FOR X=0 TO 7:POKE RAM*2
56+8+X,255:NEXT X
250 SETCOLOR 0,2,6:GOSUB 260:GOTO 320
260 ? #6;"score=0":POKE 53278,2:POSITION
 12,0:? #6;"balls=3"
270 COLOR 129:PLOT 1,4:DRAWTO 18,4:PLOT
1,5:DRAWTO 18,5
280 COLOR 33:PLOT 1,6:DRAWTO 18,6:PLOT 1
,7:DRAWTO 18,7
290 COLOR 161:PLOT 1,8:DRAWTO 18,8:PLOT
1,9:DRAWTO 18,9
300 COLOR 1:PLOT 1,10:DRAWTO 18,10:PLOT
1,11:DRAWTO 18,11
310 COLOR 33:PLOT 0,23:DRAWTO 0,1:DRAWTO
 19,1:DRAWTO 19,23:RETURN
320 RAM=RAM-8:FOR X=512 TO 768:POKE RAM*
256+X,0:NEXT X:RAM=RAM*256
330 POKE RAM+620,255:POKE RAM+621,255
340 POKE 559,46:POKE 54279,RAM/256:POKE
53277,3
350 POKE 704,15*16+10:POKE 705,54:POKE 5
3248,128:POKE 53249,128:POKE 53256,1
360 POKE 203,128:POKE 208,128:POKE 206,2
:POKE 207,1:BALL=RAM+710
370 HIGH=INT(BALL/256):LOW=BALL-256*HIGH
380 POKE 204,LOW:POKE 205,HIGH
390 X=USR(1536)
400 IF PEEK(204)+256*PEEK(205)-RAM<666 T
HEN POKE 207,1:MODE=2:POKE 53256,0
410 IF PEEK(53260)<>0 THEN X=PEEK(203)+7
:Y=PEEK(208):POKE 207,2:POKE 206,(Y>=X)*
2+(Y<X):MODE=1
420 X=USR(1536)
430 IF PEEK(53253)=0 THEN 490
440 FOR P=22 TO 12 STEP -1:SOUND 0,P,12,
15:NEXT P:SOUND 0,0,0,0:SC=SC+10:POSITIO
N 6,0:? #6;SC
450 X=INT((PEEK(208)-55)/8)+1
```

```
460 Y=PEEK(204)+256*PEEK(205)-RAM-3:Y=IN
T((Y-664)/4)+3
470 POSITION X,Y:? #6;" "
480 POKE 207,MODE
490 POKE 53278,255:X=USR(1536)
500 IF PEEK(204)+256*PEEK(205)-RAM>756 T
HEN 520
510 GOTO 390
520 FOR P=0 TO 255 STEP 7:SOUND 0,P,10,1
5:NEXT P:SOUND 0,0,0,0
530 IF PEEK(53279)<>6 THEN 530
540 BLS=BLS-1:POSITION 12,0:? #6;"balls=
";BLS:MODE=1:POKE 53256,1
550 IF BLS=0 AND PEEK(53279)=6 THEN 550
560 IF BLS=0 THEN 640
570 POKE RAM+760,0:POKE RAM+761,0:POKE R
AM+759,0:GOTO 340
580 P=0:FOR X=1 TO 18:FOR Y=4 TO 11:LOCA
TE X,Y,O:IF O<>32 THEN P=1
590 NEXT Y:NEXT X:IF P=0 THEN BLS=BLS+1:
GOSUB 260:POSITION 12,0:? #6;"balls=";BL
S
600 GRAPHICS 1+16:GOSUB 260:POKE 756,RAM
/256+8:SETCOLOR 0,2,6:POKE 559,46
610 IF PEEK(53279)=6 THEN 610
620 POKE 53279,0:IF PEEK(53279)<>6 THEN
620
630 POKE RAM+760,0:POKE RAM+761,0:POKE R
AM+759,0:GOTO 340
640 FOR X=0 TO 8:POSITION 6+X,12:? #6;G$
(X+1,X+1):FOR Y=10*X TO 10*X+9:SOUND 0,Y
,12,15
650 SETCOLOR 0,X,8:NEXT Y:NEXT X:SOUND 0
,0,0,0
660 IF PEEK(53279)<>6 THEN 660
670 SC=0:BLS=3:SOUND 0,0,0,0:GOTO 600
```

GRAND PRIX

Use your joystick to control your car left or right down the narrow lane at night.

```
10 REM ** GRAND PRIX - P.BUNN **
20 PI=5
30 X2=14:X=116:S=PEEK(106)-8:Q=S*256:FOR
 N=Q+512 TO Q+640:POKE N,0:NEXT N:POKE 5
4279,S:CRASH=250
40 POKE 559,46:POKE 53248,X:POKE 704,216
:POKE 704,200:POKE 53256,1
50 FOR N=Q+552 TO Q+561:READ A:POKE N,A:
NEXT N
60 DATA 129,195,165,24,24,153,219,165,36
,24
70 GRAPHICS 0:SETCOLOR 2,0,0:POKE 53277,
3:POKE 559,46
80 POKE 752,1:POKE 53278,A
90 FOR P=0 TO 23:FOR O=1 TO X2:? " ";:NE
XT O:? "|      |":NEXT P
100 S=STICK(0):X=X+(S=7)*2:X=X-(S=11)*2:
POKE 53248,X
110 SOUND 0,40,10,15:SC=SC+PI:SOUND 0,0,
0,0
120 IF O THEN O=0:Z=2:GOTO 170
130 A=PEEK(53770)
140 IF A>85 AND A<170 THEN Z=2
150 IF A>170 AND X2<15 THEN Z=3:O=1
160 IF A<85 AND X2>2 THEN X2=X2-1:Z=1:O=
1
170 FOR N=1 TO X2:? " ";:NEXT N
180 IF A=192 AND  NOT I THEN ? " FINISH
":I=1:GOTO 220
190 IF Z=1 THEN ? "/      /"
200 IF Z=2 THEN ? "|      |"
210 IF Z=3 THEN ? "\      \":X2=X2+1
220 Y=PEEK(53252):IF Y<>0 AND I THEN GOT
O 320
230 IF Y<>0 THEN GOTO CRASH
240 GOTO 100
250 REM ** YOU CRASHED! **
260 N=INT(RND(0)*10):POKE Q+552+N,PEEK(5
3770):SOUND 0,RND(0)*20+20,80,15:POKE 70
4,PEEK(53770)
270 IF PEEK(53770)<240 THEN 260
```

```
280 POKE 53248,0:? :? "YOU SCORED:";SC:?
 :? :? "HIT ANY KEY.":POKE 764,255
290 SOUND 0,0,0,0
300 IF PEEK(764)=255 THEN 300
310 RUN
320 FOR Y=0 TO 255:POKE 710,Y:NEXT Y
330 ? "}YOUR SCORE :";SC:POKE 710,0
340 ? "      BONUS  :1000":B=1000
350 FOR G=1 TO 1000 STEP 10:B=B-10:SC=SC
+10:POSITION 14,0:? SC:POSITION 14,1:? B
;"  ":SOUND 0,G/4,10,10:NEXT G
360 SOUND 0,0,0,0
370 ? :PI=PI*2:? "STEP VALUE NOW IS ";PI
380 RESTORE :I=0
390 GOTO 30
```

PATTERN GENERATOR

Although not a games program, this program produces very pretty patterns. Just input any positive number when prompted and see that corresponding pattern.

```
10 REM *** PATTERNS BY P.BUNN ***
20 REM THESE STEPS GIVE GOOD RESULTS:-
30 REM 121,1234,2
40 REM
50 DIM A$(10):DEG :U=1:GRAPHICS 8:COLOR
1:SETCOLOR 2,5,4:SETCOLOR 4,5,4
60 ? "WHAT STEP ";:INPUT P
70 L=L+P
80 A=(SIN(L)*70)+120:B=(COS(L)*70)+80
90 LOCATE A,B,I:IF I<>0 THEN DRAWTO A,B:
SOUND 0,0,0,0:GOTO 130
100 IF U THEN PLOT A,B:U=0
110 SOUND 0,B,10,8
120 DRAWTO A,B:NEXT L:GOTO 70
130 ? "}ANOTHER PATTERN ";:INPUT A$
140 IF A$(1,1)="Y" THEN RUN
150 IF A$(1,1)<>"N" THEN 130
160 GRAPHICS 0:END
```

ALIEN BOMBS

In this program you have to shoot down ten descending alien bombs. Use keys "1", "2" and "3" to correspond to the top, middle and lower laser cannon. Full instructions are included in the listing.

```
10 REM ***   ALIEN BOMBS   ***
20 REM
30 REM BY P.BUNN   DEC 1981
40 REM
50 DIM A$(50)
60 SETCOLOR 1,2,4:SETCOLOR 2,2,10:SETCOL
OR 4,2,10:POKE 82,1
70 TRAP 70:? "}▶ INSTRUCTIONS (Y/N)";:IN
PUT A$:IF A$<>"N" AND A$<>"Y" THEN 70
80 IF A$="N" THEN 300
90 A$=" AN EVIL WARLORD HAS SENT TEN RAD
IO ":GOSUB 360
100 A$="CONTROLLED BOMBS,AIMED AT YOUR P
LANET":GOSUB 360
110 A$=" ":GOSUB 360
120 A$="YOUR MISSION :":GOSUB 360
130 A$="   TO DESTROY ALL ALIEN BOMBS.":
GOSUB 360
140 A$="FAILURE TO COMPLETE YOUR TASK WI
LL":GOSUB 360
150 A$="RESULT IN YOUR COURT MARTIAL!":G
OSUB 360
160 A$=" ":GOSUB 360
170 A$="YOUR CONTROLS :":GOSUB 360
180 A$="  1. FIRE UPPER LASER CANNON":GO
SUB 360
190 A$="  2. FIRE MIDDLE LASER CANNON":G
OSUB 360
200 A$="  3. FIRE LOWER LASER CANNON":GO
SUB 360
210 A$=" ":GOSUB 360:A$="SCORING:":GOSUB
 360
220 A$="HIT USING UPPER CANNON =10 POINT
S":GOSUB 360
230 A$="HIT USING MIDDLE CANNON=20 POINT
S":GOSUB 360
240 A$="HIT USING LOWER CANNON =30 POINT
S":GOSUB 360:A$=" ":GOSUB 360
250 A$="ONLY A DIRECT HIT WILL PENETRATE
 THE":GOSUB 360
```

```
260 A$="ALIEN BOMB..SO BE CAREFULL!":GOS
UB 360:A$=" ":GOSUB 360
270 A$=" ":GOSUB 360:A$="  THE GAME IS O
VER WHEN EITHER:-":GOSUB 360
280 A$="1. YOU DESTROY ALL BOMBS":GOSUB
360
290 A$="2. A BOMB DESTROYS YOUR PLANET":
GOSUB 360
300 A$=" ":GOSUB 360:A$="   PRESS 'RETUR
N' TO CONTINUE":GOSUB 360
310 POKE 764,255
320 IF PEEK(764)<>12 THEN 320
330 POKE 764,255:POKE 752,1
340 FOR G=0 TO 20:POSITION 0,G:? "
                                   ":NEXT
 G
350 GOTO 370
360 FOR S=1 TO LEN(A$):FOR Q=1 TO 10:NEX
T Q:SOUND 1,120,10,12:? A$(S,S);:SOUND 1
,0,0,0:NEXT S:? :RETURN
370 REM SET UP PLANET
380 GRAPHICS 7:SETCOLOR 2,12,4:SETCOLOR
4,7,0:SETCOLOR 1,13,12
390 COLOR 3:FOR L=0 TO 159 STEP 2
400 Q=INT(RND(0)*7)+1
410 PLOT L,79:DRAWTO L,(79-Q)
420 PLOT L+1,79:DRAWTO L+1,(79-Q):NEXT L
430 SETCOLOR 0,1,8:COLOR 1
440 PLOT 2,0:DRAWTO 2,79
450 PLOT 3,10:PLOT 3,11:PLOT 4,11:PLOT 4
,12:PLOT 5,12:PLOT 4,13:PLOT 3,13:PLOT 3
,14
460 PLOT 3,30:PLOT 3,31:PLOT 4,31:PLOT 4
,32:PLOT 5,32:PLOT 4,33:PLOT 3,33:PLOT 3
,34
470 PLOT 3,50:PLOT 3,51:PLOT 4,51:PLOT 4
,52:PLOT 5,52:PLOT 4,53:PLOT 3,53:PLOT 3
,54
480 POKE 752,1:FOR L=1 TO 12:? "  GET RE
ADY!":SOUND 1,120,10,14:FOR J=1 TO 20:NE
XT J
```

```
490 ? "▶▶GET READY!":SOUND 1,70,10,14:FO
R J=1 TO 20:NEXT J
500 NEXT L
510 ? "▶▶GO..GO..GO!!!"
520 FOR L=1 TO 3:FOR Q=1 TO 80 STEP 10:S
OUND 1,Q*L,10,L+12:A=A+1
530 NEXT Q:NEXT L
540 ? "}":SOUND 1,0,0,0:S=0
550 REM MAIN PROGRAM LOOP
560 FOR OU=1 TO 10
570 R=80
580 FOR K=2 TO 76 STEP 5
590 POKE 656,2:? "SCORE:";S;"  ALIEN NO.
";OU
600 SOUND 1,RND(0)*8,120,7
610 W=INT(RND(0)*4)-2:Q=(RND(0)*5)-2:K=K
+W:R=R+Q
620 IF R>154 THEN R=154
630 IF R<4 THEN R=4
640 COLOR 1:GOSUB 770
650 P=PEEK(764):IF P=31 THEN E=10:KK=KK+
1:GOSUB 800:POKE 764,255
660 IF P=30 THEN KK=KK+1:E=20:GOSUB 840:
POKE 764,255
670 IF P=26 THEN KK=KK+1:E=30:GOSUB 880:
POKE 764,255
680 COLOR 0:GOSUB 770
690 NEXT K
700 COLOR 1:GOSUB 770:FOR L=1 TO 50
710 A=INT(RND(0)*3)+1:COLOR A:PLOT R,K:D
RAWTO RND(0)*159,RND(0)*79:SOUND 1,RND(0
)*90,120,15:NEXT L:SOUND 1,0,0,0
720 GRAPHICS 0:? "YOU LOST:..."
730 ? :? "YOUR SCORE WAS ";S
740 ? :? "YOU WILL BE DISMISSED FROM THE
"
750 ? "FEDERATION !...AND SHOT!"
760 ? :? :GOTO 1080
770 PLOT R+1,K+1:DRAWTO R+6,K+1:PLOT R,K
+2:PLOT R+1,K+2:PLOT R+3,K+2
```

```
780 PLOT R+4,K+2:PLOT R+6,K+2:PLOT R+7,K
+2:PLOT R+1,K+3:DRAWTO R+6,K+3
790 RETURN
800 COLOR 2:PLOT 6,12:DRAWTO R+5,12:COLO
R 0:PLOT 6,12:DRAWTO 159,12:FOR L=1 TO 2
55 STEP 30:SOUND 1,L,10,15:NEXT L
810 IF K>=9 AND K<=11 THEN 920
820 SOUND 1,0,0,0
830 RETURN
840 COLOR 2:PLOT 6,32:DRAWTO R+5,32:COLO
R 0:PLOT 6,32:DRAWTO 159,32:FOR L=255 TO
 1 STEP -30:SOUND 1,L,10,15:NEXT L
850 IF K>=29 AND K<=31 THEN 920
860 SOUND 1,0,0,0
870 RETURN
880 COLOR 2:PLOT 6,52:DRAWTO R+5,52:COLO
R 0:PLOT 6,52:DRAWTO 159,52:FOR L=155 TO
 355 STEP 30:SOUND 1,L,10,15:NEXT L
890 IF K>=49 AND K<=51 THEN 920
900 SOUND 1,0,0,0
910 RETURN
920 REM EXPLOSION!
930 POP :POKE 764,255
940 FOR N=1 TO 10
950 Q=(RND(0)*12)-5:W=(RND(0)*12)-5
960 COLOR INT(RND(0)*3)+1
970 PLOT R+3,K+3:DRAWTO R+Q,K+W:SOUND 1,
RND(0)*90,120,15:NEXT N
980 COLOR 0
990 FOR L=R-7 TO R+7:PLOT L,K-7:DRAWTO L
,K+7:NEXT L:S=S+E:SOUND 1,0,0,0
1000 E=0
1010 NEXT OU
1020 FOR G=1 TO 3:FOR J=1 TO 255 STEP 4:
SOUND 1,J,10,15:NEXT J:NEXT G
1030 SOUND 1,0,0,0
1040 GRAPHICS 0:? :? :? "YOU DID IT !!!"
1050 ? :? "YOU KILLED EVERY SINGLE ALIEN
 BOMB "
1060 ? :? "YOU USED ";KK;" SHOTS OF YOUR
```

```
 AMMO"
1070 ? :? "THAT GIVES YOU A SCORE OF ";S
-(KK*2)
1080 ? :? "ANOTHER GAME ";:TRAP 1080:INP
UT A$
1090 IF A$(1,1)="Y" THEN RUN
1100 ? :? :? :? "THANKS FOR THE GAME!":?
 :? :END
```

TRAP 'EM

In this exciting one or two player game, you have to try and surround your opponent. This is done by using the two joysticks to force your opponent to crash. The player to force his/her opponent to crash ten times is the winner.

There is also the availability of playing in two different sizes of blocks, large or small. The computer will ask you at the beginning of the game as to which size blocks you would like to play with. Just respond 'l' for large or 's' for small.

There is also the feature to play against the computer, Arnold. The computer plays quite well but is easily beaten with small blocks. Playing the computer using large blocks presents a real challenge.

```
10 REM ** Trap'em   Written By **
20 REM ** Paul Bunn , FEB 1982 **
30 REM **************************
40 DIM A$(15),B$(15),SL$(6)
50 GRAPHICS 0:SETCOLOR 4,8,2:SETCOLOR 2,
8,2:? "WHAT IS PLAYER 1'S NAME ";:INPUT
A$
60 ? "DO YOU WISH TO PLAY AGAINST THE
    COMPUTER ";:INPUT B$:IF B$(1,1)="Y"
THEN CMP=1
70 ? :IF  NOT CMP THEN ? "OK, PLAYER 2'S
 NAME ";:INPUT B$:GOTO 90
80 ? "OK , YOUR GOING TO PLAY ARNOLD.":B
$="ARNOLD"
90 ? :? "  THANK YOU."
100 ? "DO YOU WISH TO PLAY WITH SMALL OR
     LARGE BLOCKS ";:INPUT SL$
110 ? :? :? "HIT THE RED BUTTON TO CONTI
NUE."
120 IF STRIG(0)=1 AND STRIG(1)=1 THEN 12
0
130 IF SL$(1,1)="L" THEN GRAPHICS 19:X1=
7:Y1=10:X2=33:Y2=10
140 IF SL$(1,1)="S" THEN GRAPHICS 21:X1=
12:Y1=24:Y2=24:X2=68
150 E1=1:F3=1:E2=0:E3=0:E4=0:F1=0:F2=0:F
4=0
160 COLOR 3
170 IF SL$(1,1)="L" THEN PLOT 2,2:DRAWTO
 38,2:DRAWTO 38,23:DRAWTO 2,23:DRAWTO 2,
2
180 IF SL$(1,1)="S" THEN PLOT 2,2:DRAWTO
 78,2:DRAWTO 78,46:DRAWTO 2,46:DRAWTO 2,
2
190 COLOR 1:PLOT X1,Y1
200 COLOR 2:PLOT X2,Y2
210 S1=STICK(0):S2=STICK(1)
220 IF S1=15 THEN 320
230 SOUND 0,90,10,8
240 IF S1=7 OR S1=6 OR S1=5 THEN E1=1:GO
TO 260
```

```
250 E1=0
260 IF S1=5 OR S1=13 OR S1=9 THEN E2=1:G
OTO 280
270 E2=0
280 IF S1=9 OR S1=11 OR S1=10 THEN E3=1:
GOTO 300
290 E3=0
300 IF S1=10 OR S1=14 OR S1=6 THEN E4=1:
GOTO 320
310 E4=0
320 SOUND 0,130,10,8:IF S2=15 AND  NOT C
MP THEN 510
330 IF  NOT CMP THEN 430
340 IF RND(0)>0.91 THEN R=INT(RND(0)*4)+
1:II=1:IF R=1 THEN LOCATE X2+1,Y2,J:IF J
=0 THEN F1=1:F2=0:F3=0:F4=0
350 IF II AND R=2 THEN LOCATE X2,Y2+1,J:
IF J=0 THEN F1=0:F2=1:F3=0:F4=0
360 IF II AND R=3 THEN LOCATE X2-1,Y2,J:
IF J=0 THEN F1=0:F2=0:F3=1:F4=0
370 IF II AND R=4 THEN LOCATE X2,Y2-1,J:
IF J=0 THEN F1=0:F2=0:F3=0:F4=1
380 IF F3=1 THEN LOCATE X2-1,Y2,K:IF K=0
 THEN 510
390 IF F1=1 THEN LOCATE X2+1,Y2,K:IF K=0
 THEN 510
400 IF F2=1 THEN LOCATE X2,Y2+1,K:IF K=0
 THEN 510
410 IF F4=1 THEN LOCATE X2,Y2-1,K:IF K=0
 THEN 510
420 II=0:GOTO 930
430 IF S2=7 OR S2=6 OR S2=5 THEN F1=1:GO
TO 450
440 F1=0
450 IF S2=5 OR S2=13 OR S2=9 THEN F2=1:G
OTO 470
460 F2=0
470 IF S2=9 OR S2=11 OR S2=10 THEN F3=1:
GOTO 490
480 F3=0
490 IF S2=10 OR S2=14 OR S2=6 THEN F4=1:
```

```
GOTO 510
500 F4=0
510 SOUND 0,0,0,0
520 IF E1=1 THEN X1=X1+1
530 IF E2=1 THEN Y1=Y1+1
540 IF E3=1 THEN X1=X1-1
550 IF E4=1 THEN Y1=Y1-1
560 IF F1=1 THEN X2=X2+1
570 IF F2=1 THEN Y2=Y2+1
580 IF F3=1 THEN X2=X2-1
590 IF F4=1 THEN Y2=Y2-1
600 REM ** CHECK IF CRASH **
610 LOCATE X1,Y1,C1
620 LOCATE X2,Y2,C2
630 IF C1<>0 OR C2<>0 THEN 650
640 GOTO 190
650 FOR H=1 TO 30:FOR G=60 TO 80 STEP 6
660 IF C1<>0 THEN COLOR H:PLOT X1,Y1
670 IF C2<>0 THEN COLOR H:PLOT X2,Y2
680 SOUND 0,G,10,10
690 NEXT G:NEXT H
700 GRAPHICS 0:SOUND 0,0,0,0
710 ? "  ";:SETCOLOR 2,8,2:SETCOLOR 4,8,2
720 IF C1<>0 AND C2<>0 THEN ? A$;" AND "
;B$;:GOTO 770
730 IF C2<>0 THEN ? B$;:SC1=SC1+1
740 IF C1<>0 THEN ? A$;:SC2=SC2+1
750 REM ** Someone has crashed **
760 ? " BITES THE DUST.":GOTO 780
770 ? " BITE THE DUST.":SC1=SC1+1:SC2=SC
2+1
780 ? :?
790 ? "  SCORES :-"
800 ? A$;":";SC1;"    ";B$;":";SC2
810 IF SC1=10 OR SC2=10 THEN 880
820 ? :?
830 IF SC1>=SC2+5 THEN ? A$;"'S WINNING
BY FAR !!!":GOTO 870
840 IF SC2>=SC1+5 THEN ? B$;"'S WINNING
```

```
BY FAR !!!":GOTO 870
850 IF SC1=SC2 THEN ? "IT'S A DRAW AT TH
E MOMENT...":GOTO 870
860 ? "THERE'S NOT MUCH IN IT...."
870 GOTO 110
880 ? :? :? :FOR G=1 TO 255 STEP 2:SOUND
 0,G,10,10:NEXT G:FOR G=255 TO 0 STEP -6
:SOUND 0,G,10,10:NEXT G:SOUND 0,0,0,0
890 IF SC1>SC2 THEN ? A$;" WINS THE GAME
!...":? "CONGRATULATIONS ";A$:GOTO 920
900 IF SC2>SC1 THEN ? B$;" WINS THE GAME
!...":? "CONGRATULATIONS ";B$:GOTO 920
910 ? "WELL DONE , BOTH OF YOU....":? "I
T'S A DRAW !!!"
920 END
930 REM *ARNOLD'S PATH IS BLOCKED*
940 IF F1=1 THEN LOCATE X2,Y2-1,L1:IF L1
=0 THEN F4=1:F3=0:F2=0:F1=0:GOTO 510
950 IF F1=1 THEN F2=1:F1=0:F3=0:F4=0:GOT
O 510
960 IF F2=1 THEN LOCATE X2-1,Y2,L1:LOCAT
E X2+1,Y2,L2:IF L1=0 THEN F4=0:F3=1:F2=0
:F1=0:GOTO 510
970 IF F2=1 THEN F2=0:F1=1:F3=0:F4=0:GOT
O 510
980 IF F3=1 THEN LOCATE X2,Y2-1,L1:IF L1
=0 THEN F4=1:F3=0:F2=0:F1=0:GOTO 510
990 IF F3=1 THEN F2=1:F1=0:F3=0:F4=0:GOT
O 510
1000 IF F4=1 THEN LOCATE X2+1,Y2,L1:IF L
1=0 THEN F4=0:F3=0:F2=0:F1=1:GOTO 510
1010 IF F4=1 THEN F2=0:F1=0:F3=1:F4=0:GO
TO 510
```

REACTION TIMER

This short program uses a short machine code subroutine to time how long it takes you to press the space bar. If you cheat and hold the space bar down, the computer will give the worst time possible: 65535.

To play, type RUN and then as soon as you see an orange-coloured dot appear on the screen, press the space bar. Sounds simple? Then try and get a time of less than 16,000!!!

```
10 REM *** REACTION TIMER      ***
20 REM *** WRITTEN FEB 1982    ***
30 REM *** BY PAUL BUNN.       ***
40 FOR L=0 TO 37:READ A:POKE 1535+L,A:NE
XT L
50 DATA 104,173,252,2,24,201,33,240,21,1
69,0,133,213,133,212,173,252,2,201,33
60 DATA 240,15,230,212,208,245,230,213,2
08,241,169,255,133,213
70 DATA 133,212,96,96
80 GRAPHICS 0
90 ? "TO PLAY : WHEN YOU SEE A ORANGE SP
OT  IN THE MIDDLE OF THE SCREEN , PRESS"
100 ? "THE SPACE BAR AS QUICKLY AS YOU C
AN."
110 OPEN #1,4,0,"K:"
120 ? "                READY ? ? ?"
130 GET #1,A:IF A<>ASC("Y") THEN 130
140 ? "}"
150 GRAPHICS 19:COLOR 1
160 FOR L=1 TO INT(RND(0)*1000)+600:NEXT
 L
170 PLOT 20,12:POKE 212,0:POKE 213,0:X=U
SR(1535)
180 GRAPHICS 0:? :? "SCORE : ";X
190 IF X=65535 THEN ? "DID YOU CHEAT ?":
GOTO 260
200 IF X<9200 THEN ? "WHERE DID YOU GET
YOUR LIGHTNING FAST REACTION ?":GOTO 260
210 IF X<15000 THEN ? "EXCELLENT.":GOTO
260
220 IF X<19000 THEN ? "VERY GOOD INDEED.
":GOTO 260
230 IF X<23000 THEN ? "NOT BAD.":GOTO 26
0
240 IF X<28000 THEN ? "PATHETIC!":GOTO 2
60
250 ? "THAT WAS TERRIBLE!!!.........."
260 ? :? :? "ANOTHER TRY ? (Y-N)"
270 GET #1,A:IF A=ASC("Y") THEN 150
```

```
280 IF A<>ASC("N") THEN 270
290 FOR G=1 TO 9:? :NEXT G:? "  GOOD BYE
.":END
```

REVERSI

Reversi was invented in 1888 and is played on a standard draughts board, using double-sided pieces - white on one side, black on the other. In his splendid book "Discovering old Board Games" (Shire Publications Ltd, Aylesbury, 1980), R.C. Bell explains that black begins the game by placing a piece, blackside up on one of the four central squares on the empty board. White replies by placing his/her first piece whiteside up on another central square. "These four squares are covered in the first four turns of play, and then the players continue alternately, placing their pieces on a square adjacent to one occupied by an enemy piece", Mr. Bell writes. Any enemy piece in a straight line between the latest piece played and another one of the player's pieces when the board is completely covered, or when neither player can move.

You'll find that the computer is very hard to beat but plays quite slowly. To enter your move as a single double digit number corresponding to the number of squares across by the number of squares down. e.g. 45 would be 4 across, and 5 down. Happy playing!

```
10 REM * * *        REVERSI        * * *
20 DIM A(10,10)
30 FOR B=1 TO 10:FOR C=1 TO 10
40 IF B<>1 AND C<>1 AND B<>10 AND C<>10
THEN A(B,C)=46
50 NEXT C:NEXT B
60 A(5,5)=88:A(6,6)=88:A(6,5)=79:A(5,6)=
79:R=0
70 GRAPHICS 7:POKE 752,1:SETCOLOR 4,12,2
:SETCOLOR 2,12,2:SETCOLOR 1,0,12:SETCOLO
R 0,0,0:OPEN #1,4,0,"K:"
80 POSITION 2,11:POKE 752,1:? "Would you
 like to go first (Y/N)  ?"
90 GET #1,A:IF A<>89 AND A<>78 THEN 90
100 GOSUB 440
110 IF A=89 THEN 350
120 S=79:T=88:H=0
130 FOR A=2 TO 9:FOR B=2 TO 9:IF A(A,B)<
>46 THEN 290
140 Q=0:FOR C=-1 TO 1:FOR D=-1 TO 1:K=0:
F=A:G=B
150 IF A(F+C,G+D)<>S THEN 170
160 K=K+1:F=F+C:G=G+D:GOTO 150
170 IF A(F+C,G+D)<>T THEN 190
180 Q=Q+K
190 NEXT D
200 NEXT C
210 IF A=2 OR A=9 THEN Q=Q*2
220 IF B=2 OR B=9 THEN Q=Q*2
230 IF B=3 OR B=8 THEN Q=Q/2
240 IF A=3 OR A=8 THEN Q=Q/2
250 IF (A=2 OR A=9) AND (B=3 OR B=8) THE
N Q=Q/2
260 IF (A=3 OR A=8) AND (B=2 OR B=9) THE
N Q=Q/2
270 IF Q<H OR Q=0 OR (RND(0)>0.3 AND Q=H
) THEN 290
280 H=Q:M=A:N=B
290 NEXT B
300 NEXT A
```

```
310 IF H=0 AND R=0 THEN 610
320 IF H=0 THEN 340
330 GOSUB 540
340 GOSUB 440
350 ? "What is your move (enter '0' if y
ou   can not go) ";
360 S=88:T=79
370 TRAP 370:INPUT R:TRAP 40000
380 IF R=0 THEN 420
390 IF R<11 OR R>88 OR R<>INT(R) THEN 35
0
400 M=INT(R/10)+1:N=R-10*INT(R/10)+1
410 GOSUB 540
420 GOSUB 440
430 GOTO 120
440 REM * * * * DRAW BOARD * * * *
450 COLOR 1:FOR X=44 TO 116 STEP 9:PLOT
 44,X-44:DRAWTO 116,X-44:PLOT X,0:DRAWTO
 X,72:NEXT X
460 FOR X=2 TO 9:FOR Y=2 TO 9
470 IF A(X,Y)<>46 THEN GOSUB 650
480 NEXT Y:NEXT X
490 S1=0:S2=S1:FOR X=2 TO 9:FOR Y=2 TO 9
:IF A(X,Y)=79 THEN S1=S1+1
500 IF A(X,Y)=88 THEN S2=S2+1
510 NEXT Y:NEXT X
520 ? :? "SCORES :-     ME  ";S2;"   YOU
 ";S1
530 RETURN
540 FOR C=-1 TO 1:FOR D=-1 TO 1:F=M:G=N
550 IF A(F+C,G+D)<>S THEN 570
560 F=F+C:G=G+D:GOTO 550
570 IF A(F+C,G+D)<>T THEN 600
580 A(F,G)=T:IF M=F AND N=G THEN 600
590 F=F-C:G=G-D:GOTO 580
600 NEXT D:NEXT C:RETURN
610 IF S2>S1 THEN ? "I WON ";S2;"-";S1
620 IF S2<S1 THEN ? "YOU WON ";S1;"-";S2
630 IF S1=S2 THEN ? "IT'S A DRAW"
```

```
640 END
650 COLOR (A(X,Y)=79)*1+(A(X,Y)=88)*2
660 P=((X-1)*9)+40:O=((Y-1)*9)-4
670 PLOT P-1,O-4:PLOT P,O-4
680 PLOT P-2,O-3:DRAWTO P+1,O-3
690 PLOT P-3,O-2:DRAWTO P+2,O-2
700 PLOT P-4,O-1:DRAWTO P+3,O-1
710 PLOT P-4,O:DRAWTO P+3,O
720 PLOT P-3,O+1:DRAWTO P+2,O+1
730 PLOT P-2,O+2:DRAWTO P+1,O+2
740 PLOT P-1,O+3:PLOT P,O+3
750 RETURN
```

SALES ANALYSIS

This simple business program was designed for shopowners, so that they could see how sales were going for a particular commodity. The program can be used to analyse up to eight goods or services. You input the number of sales for each month for each goods/service. The number of sales must not be above 84.

After all data has been entered, you will be able to see the data on the screen so that you can alter anything if you so desire, say a mistake on data entry. The data will then be displayed visually, in the form of a graph. The red line indicates the average.

```
10 REM ** SALES ANALYSIS **
20 GRAPHICS 0:POSITION 11,0:? "SALES ANA
LYSIS"
30 POSITION 17,1:? "BY"
40 POSITION 10,2:? "PAUL BUNN IN 1982"
50 DIM M$(12*9),T$(15),Q(12):M$(1)=" ":M
$(108)=" ":M$(2)=M$
60 FOR M=0 TO 11:READ T$,J:M$(M*9+1,M*9+
9)=T$:Q(M+1)=J:NEXT M
70 DATA JANUARY,7,FEBUARY,7,MARCH,5,APRI
L,5,MAY,3,JUNE,4,JULY,4,AUGUST,6,SEPTEMB
ER,9,OCTOBER,7,NOVEMBER,8
80 DATA DECEMBER,8
90 OPEN #1,4,0,"K:"
100 POKE 764,60
110 ? :? :? "Press 'RETURN' when ready."
120 GET #1,A:IF A<>155 THEN 110
130 ? :?
140 ? "How many goods/sevices are sold "
;:TRAP 140:INPUT I:TRAP 40000
150 IF I<>INT(I) OR I<1 OR I>8 THEN ? "
BETWEEN 1 & 8":GOTO 140
160 DIM GS$(15*I):GS$(1)=" ":GS$(15*I)="
 ":GS$(2)=GS$
170 FOR P=1 TO I
180 ? "Good ";P;"=";:TRAP 180:INPUT T$:T
RAP 40000
190 GS$((P-1)*15+1,(P-1)*15+14)=T$
200 NEXT P
210 ? :? :? "Thank you..."
220 DIM SALES(12,I),PRSNT(I)
230 ? :? "Press 'RETURN' to continue."
240 GET #1,A:IF A<>155 THEN 240
250 FOR S=1 TO I:? "}Sales analysis of "
;GS$((S-1)*15+1,(S-1)*14+15)
260 TRAP 260:? "Input last years avarage
 sale (enter  '0' if not known)";:INPUT
A:PRSNT(S)=A
270 FOR M=1 TO 12
280 ? "}  ";M$((M-1)*9+1,(M-1)*9+9)
```

```
290 ? "  ";:FOR P=1 TO Q(M):? " ";:NEXT
P:? :?
300 REM ** Analyse sales **
310 ? "Sales analysis of ";GS$((S-1)*15+
1,(S-1)*14+15)
320 TRAP 320:? :? "TOTAL SALES :";:INPUT
 P:SALES(M,S)=P
330 IF P>84 THEN ? "MAXIMUM NO. OF SALES
 IS 84!":GOTO 320
340 TRAP 40000
350 NEXT M:NEXT S
360 ? :? :? "}Sales analysis complete."
370 ? :? "Would you like to check ";:TRA
P 370:INPUT T$:IF T$(1,1)="n" THEN 500
380 FOR P1=1 TO I:? "}":FOR P2=1 TO 12
390 ? P2;")";M$((P2-1)*9+1,(P2-1)*9+9),"
SALES=";SALES(P2,P1)
400 NEXT P2
410 TRAP 430:? "Alter anything (enter nu
mber)";:INPUT T:TRAP 40000
420 IF T>0 AND T<13 THEN GOTO 450
430 NEXT P1
440 GOTO 500
450 ? "Enter new sales for ";M$((T-1)*9+
1,(T-1)*9+9);
460 TRAP 460:INPUT W:IF W<0 OR W>84 THEN
 ? "Between 1 and 84!":GOTO 460
470 TRAP 40000
480 SALES(T,P1)=W
490 GOTO 410
500 ? :? :? " Now to see graphs."
510 GOTO 1470
520 REM **DRAW GRAPH SUBROUTINE**
530 RESTORE :FOR W=1 TO 24:READ T$:NEXT
W
540 COLOR 1:PLOT 5,0:DRAWTO 5,83:DRAWTO
159,83
550 FOR K=78 TO 4 STEP -4
560 PLOT 2,K:DRAWTO 4,K:NEXT K
570 FOR K=13 TO 159 STEP 13:PLOT K,84:DR
```

```
AWTO K,86
580 FOR I1=88 TO 94:FOR I2=-3 TO 3
590 READ A:IF A=1 THEN PLOT K+I2,I1
600 NEXT I2:NEXT I1
610 NEXT K
620 DATA 0,0,0,1,0,0,0
630 DATA 0,0,0,1,0,0,0
640 DATA 0,0,0,1,0,0,0
650 DATA 0,0,0,1,0,0,0
660 DATA 0,0,0,1,0,0,0
670 DATA 1,0,0,1,0,0,0
680 DATA 0,1,1,0,0,0,0
690 DATA 1,1,1,1,1,1,0
700 DATA 1,0,0,0,0,0,0
710 DATA 1,0,0,0,0,0,0
720 DATA 1,0,0,0,0,0,0
730 DATA 1,1,1,1,0,0,0
740 DATA 1,0,0,0,0,0,0
750 DATA 1,0,0,0,0,0,0
760 DATA 1,1,0,0,0,1,1
770 DATA 1,0,1,0,1,0,1
780 DATA 1,0,0,1,0,0,1
790 DATA 1,0,0,1,0,0,1
800 DATA 1,0,0,0,0,0,1
810 DATA 1,0,0,0,0,0,1
820 DATA 1,0,0,0,0,0,1
830 DATA 0,0,0,1,0,0,0
840 DATA 0,0,1,0,1,0,0
850 DATA 0,1,0,0,0,1,0
860 DATA 0,1,1,1,1,1,0
870 DATA 0,1,0,0,0,1,0
880 DATA 0,1,0,0,0,1,0
890 DATA 0,1,0,0,0,1,0
900 DATA 1,1,0,0,0,1,1
910 DATA 1,0,1,0,1,0,1
920 DATA 1,0,0,1,0,0,1
930 DATA 1,0,0,1,0,0,1
940 DATA 1,0,0,0,0,0,1
```

```
950 DATA 1,0,0,0,0,0,1
960 DATA 1,0,0,0,0,0,1
970 DATA 0,0,0,1,0,0,0
980 DATA 0,0,0,1,0,0,0
990 DATA 0,0,0,1,0,0,0
1000 DATA 0,0,0,1,0,0,0
1010 DATA 0,0,0,1,0,0,0
1020 DATA 1,0,0,1,0,0,0
1030 DATA 0,1,1,0,0,0,0
1040 DATA 0,0,0,1,0,0,0
1050 DATA 0,0,0,1,0,0,0
1060 DATA 0,0,0,1,0,0,0
1070 DATA 0,0,0,1,0,0,0
1080 DATA 0,0,0,1,0,0,0
1090 DATA 1,0,0,1,0,0,0
1100 DATA 0,1,1,0,0,0,0
1110 DATA 0,0,0,1,0,0,0
1120 DATA 0,0,1,0,1,0,0
1130 DATA 0,1,0,0,0,1,0
1140 DATA 0,1,1,1,1,1,0
1150 DATA 0,1,0,0,0,1,0
1160 DATA 0,1,0,0,0,1,0
1170 DATA 0,1,0,0,0,1,0
1180 DATA 0,1,1,1,1,1,0
1190 DATA 1,0,0,0,0,0,0
1200 DATA 1,0,0,0,0,0,0
1210 DATA 0,1,1,1,1,1,0
1220 DATA 0,0,0,0,0,0,1
1230 DATA 0,0,0,0,0,0,1
1240 DATA 0,1,1,1,1,1,0
1250 DATA 0,0,1,1,1,0,0
1260 DATA 0,1,0,0,0,1,0
1270 DATA 1,0,0,0,0,0,1
1280 DATA 1,0,0,0,0,0,1
1290 DATA 1,0,0,0,0,0,1
1300 DATA 0,1,0,0,0,1,0
1310 DATA 0,0,1,1,1,0,0
1320 DATA 1,0,0,0,0,0,1
```

```
1330 DATA 1,1,0,0,0,0,1
1340 DATA 1,0,1,0,0,0,1
1350 DATA 1,0,0,1,0,0,1
1360 DATA 1,0,0,0,1,0,1
1370 DATA 1,0,0,0,0,1,1
1380 DATA 1,0,0,0,0,0,1
1390 DATA 1,1,0,0,0,0,0
1400 DATA 1,0,1,0,0,0,0
1410 DATA 1,0,0,1,0,0,0
1420 DATA 1,0,0,0,1,0,0
1430 DATA 1,0,0,0,1,0,0
1440 DATA 1,0,0,1,0,0,0
1450 DATA 1,1,1,0,0,0,0
1460 RETURN
1470 REM ** DRAW GRAPHS **
1480 ? "sales on y-axis"
1490 ? "Months on x-axis"
1500 ? :? "red line indicates avarage."
1510 FOR G=0 TO I-1:? "Graph for sales o
f ";GS$(G*15+1,G*15+14)
1520 ? :? "ready ??"
1530 GET #1,A:IF A<>ASC("y") THEN 1530
1540 GRAPHICS 7+16:SETCOLOR 0,0,0:SETCOL
OR 4,2,8:GOSUB 520
1550 SETCOLOR 1,7,4:COLOR 2:SETCOLOR 2,3
,4
1560 PLOT 6,84-PRSNT(G+1)
1570 FOR K=13 TO 156 STEP 13
1580 DRAWTO K,84-SALES(K/13,G+1)
1590 NEXT K
1600 T=0:FOR W=1 TO 12:T=T+SALES(W,G+1):
NEXT W:AVRGE=T/12
1610 COLOR 3:PLOT 6,84-AVRGE:DRAWTO 159,
84-AVRGE
1620 GET #1,A:IF A<>155 THEN 1600
1630 NEXT G
1640 GRAPHICS 0:? "Analysis complete ":E
ND
```

OLD FORTY-NINER

This game is like checkers, except it is played on a seven by seven board (rather than eight by eight, as in checkers). The pieces move as checkers pieces–diagonally one square, jumping over an opponent for a capture, into an empty square beyond. The main difference from checkers, apart from the size of the board, is that pieces may move forward and backward at will. There are no 'kings', as every piece can move as if it is a king, and there are no multiple jumps.

The computer is the X's moving down the screen, and you are the O's. You move by entering the number of the square you're moving from (entering the number along the left hand edge first, then the number along the top, then pressing RETURN) then the number of the square you're moving from.

The computer keeps track of the score, tells you (before it moves its pieces) which square it is moving to, and terminates the game as soon as one player manages to capture five of the opponent's pieces.

The program shows some structured programming techniques. Line 30 sends the action to a subroutine at the end of the program which initialises the variables. The computer then goes to its move routine, starting at 7000. Having made a move, it returns to the start of the program, and from there goes to the subroutine at 8000 to print out the board. From there it goes to subroutine 6000 to accept the player's move, 8000 to reprint the board, then back to the computer's next move . . . and so on.

The advantage of working in this way - with a specific sub-routine for each section of the program - is that you can easily alter part of the program which appears unsatis-factory, without facing the danger of running out of lines, or of 'getting lost' in the program. Placing the variables assignment at the end has the advantage that you can add new ones, as the need for them becomes apparent, without any real problems.

```
10 REM OLE FORTY-NINER (7X7 CHECKERS)
20 REM (C) HARTNELL 1982
30 GOSUB 9000:REM INITIALISE
40 GOSUB 7000:REM COMPUTER MOVES
60 GOSUB 8000:REM PRINT BOARD
70 GOSUB 6000:REM ACCEPT PLAYER MOVE
90 GOSUB 8000:REM PRINT BOARD
1000 GOTO 40
6000 PRINT
6050 PRINT "FROM";
6060 INPUT M
6070 PRINT "TO";
6080 INPUT N
6090 H(N)=120
6095 IF ABS(M-N)=22 OR ABS(M-N)=18 THEN
H((M+N)/2)=46
6097 IF ABS(M-N)=22 OR ABS(M-N)=18 THEN
ME=ME+1
6100 H(M)=46
6990 RETURN
7000 REM COMPUTER MOVES
7010 FOR A=76 TO 12 STEP -1
7020 IF H(A)<>111 THEN 7060
7030 FOR B=1 TO 4
7032 IF A<28 AND B<3 THEN 7050
7033 IF A>60 AND B>2 THEN 7060
7035 Q=2*Z(B)
7040 IF H(A+Z(B))=120 AND H(A+Q)=46 THEN
 7070
7050 NEXT B
7060 NEXT A
7065 GOTO 7300
7070 H(A+Z(B))=46
7080 H(A)=46
7090 H(A+Q)=111
7092 Y=A+Q
7093 X=A
7095 IT=IT+1
7100 RETURN
```

```
7200 NEXT A
7300 REM RANDOM MOVE
7310 Y=0
7320 Y=Y+1
7330 K=INT(RND(1)*66)+12
7340 IF H(K)<>111 AND Y<100 THEN 7320
7350 IF H(K)<>111 THEN 7460
7360 FOR T=1 TO 4
7370 IF H(K+Z(T))=46 THEN 7400
7380 NEXT T
7390 IF Y<70 THEN 7310
7395 GOTO 7460
7400 H(K+Z(T))=111
7410 H(K)=46
7415 X=K:Y=K+Z(T)
7420 RETURN
7460 PRINT "I CONCEDE"
7470 END
7990 RETURN
8000 REM CLEAR SCREEN
8010 PRINT
8015 PRINT "I MOVED FROM ";X;" TO ";Y
8017 PRINT
8020 PRINT "SCORES: YOU: ";ME;"   ME: ";
IT
8040 PRINT :PRINT :PRINT
8050 PRINT ,"  1234567"
8055 PRINT ,"###########"
8060 FOR J=70 TO 10 STEP -10
8070 A=H(J+1):B=H(J+2):C=H(J+3):D=H(J+4)
:E=H(J+5):F=H(J+6):G=H(J+7)
8080 PRINT ,J/10;"#";CHR$(A);CHR$(B);CHR
$(C);CHR$(D);CHR$(E);CHR$(F);CHR$(G);"#"
;J/10
8090 NEXT J
8100 PRINT ,"###########"
8110 PRINT ,"  1234567"
8120 IF IT=5 OR ME=5 THEN 8140
8130 RETURN
```

```
8140 IF IT=5 THEN PRINT "I WIN!"
8150 IF ME=5 THEN PRINT "YOU WIN!"
8999 END
9000 REM INITIALISE
9010 IT=0:ME=0
9020 DIM H(99),Z(4)
9030 FOR A=1 TO 99
9040 H(A)=0
9050 IF A>77 OR A=70 OR A=60 OR
A=68 OR A=69 OR A=50 OR A=59 OR
A=58 OR A=40 OR A=49 OR A=48 THE
N GOTO 9090
9055 IF A=30 OR A=38 OR A=39 OR
A=20 OR A=28 OR A=29 OR A<11 THE
N GOTO 9090
9060 H(A)=46:REM .
9070 IF A=72 OR A=74 OR A=76 OR A=61 OR
A=63 OR A=65 OR A=67 THEN H(A)=111
9080 IF A=21 OR A=23 OR A=25 OR A=27 OR
A=12 OR A=14 OR A=16 THEN H(A)=120:REM x
9090 NEXT A
9100 FOR A=1 TO 4:READ B:Z(A)=B:NEXT A
9110 DATA -11,-9,11,9
9200 RETURN
```

GANYMEDE 1V

This game, in the HUNT THE WUMPUS genre, places you in a maze of tunnels on Ganymede 1V, where–if you survive for 25 minutes–you'll be rescued. There are beasties, gems and other things to deal with. The beasties tend to kill you, the quicksand is inescapable, the gems may make you rich (if you survive), but the space warps are the worst of all. Enter a cave containing one of these and you will be transported to another cave in the system. From time to time your computer will print out the cave system as seen from above.

Once you've got this program up and running, you can use the basic framework to create as many HUNT THE WUMPUS type games as you like, placing contents and animals of your own choice in the caves. There is also a lot of scope for adding colour, sound and graphics to the program.

The game is straightforward to play. You start in cave ('sector') 55, and can move up, down, right or left one square at a time. Your Atari will warn you when things are nearby, but due to a defect, will only tell you of the contents of one nearby cave/sector, and not of all of them. It will **not** tell which of the nearby caves it is describing. A cave is empty after you leave it, so you cannot revisit a gems cave over and over again to enrich yourself. If you enter a quicksand cave, the game ends with a printout of the whole system. The same happens if a beastie eats you.

```
10 REM * * * * * GANYMEDE IV * * * * *
20 REM (c) Hartnell 1982
30 REM
40 GRAPHICS 0:DIM G$(11),A(100),Z$(2):DL
=PEEK(560)+256*PEEK(561)+4:POKE DL-1,70:
POKE DL+2,6:POKE 752,1:E=55
50 FOR X=1 TO 11:READ Y:G$(X,X)=CHR$(Y):
NEXT X:DATA 71,97,238,89,109,229,68,101,
32,233,86
60 SETCOLOR 1,6,10:SETCOLOR 2,0,0:SETCOL
OR 3,11,10:OPEN #1,4,0,"K:":SETCOLOR 0,1
4,10
70 FOR X=1 TO 11:POSITION X+3,0:? G$(X,X
)
80 FOR Y=15 TO 0 STEP -1:SOUND 0,X*20,8,
Y:NEXT Y:GOSUB 100:NEXT X
90 GOTO 120
100 A=PEEK(708):POKE 708,PEEK(709):POKE
709,PEEK(711):POKE 711,A:RETURN
110 ? CHR$(125):POSITION 4,0:? G$:RETURN

120 FOR X=1 TO 100:A(X)=46:IF X<12 OR X>
90 THEN A(X)=47
130 Z$=STR$(X*10):IF Z$(2)="0" OR Z$(2)=
"1" THEN A(X)=47
140 GOSUB 100:SOUND 0,(INT(X/3)<>X/3)*X,
8,8:NEXT X:SOUND 0,0,0,0
150 FOR X=1 TO 6:A(INT(RND(0)*76+12))=47
:A(INT(RND(0)*76+12))=63:A(INT(RND(0)*76
+12))=66
160 A(INT(RND(0)*76+12))=81:A(INT(RND(0)
*76+12))=71:GOSUB 100:SOUND 0,X*10,12,8:
NEXT X:SOUND 0,0,0,0
170 FOR X=255 TO 0 STEP -5:SOUND 0,X,12,
15:GOSUB 100:NEXT X:SOUND 0,0,0,0
180 A(E)=72:Q=INT(RND(0)*7):IF Q=0 THEN
GOSUB 1040
190 Q=1:? "You are in cave number ";E:IF
 G>0 THEN ? "With Ganymede gems worth $"
;G
200 GOSUB 800:? :? :? "You must survive
for another ";25-H:? "more minutes!":?
```

```
210 ? :? :? "Which direction to you want
 to move"
220 ? "(North,South,East or West) ?";:PO
KE 764,255
230 IF PEEK(764)<>255 THEN GET #1,Z:Z$=C
HR$(Z):X=0:X=X+(Z$="W")+(Z$="S")+(Z$="E"
)+(Z$="N"):IF X=1 THEN 250
240 GOSUB 100:FOR X=1 TO 30:NEXT X:GOTO
230
250 U=0:POKE 764,255:IF Z$="N" THEN ? "N
ORTH"
260 IF Z$="S" THEN ? "SOUTH"
270 IF Z$="E" THEN ? "EAST"
280 IF Z$="W" THEN ? "WEST"
290 FOR X=15 TO 0 STEP -0.4:SOUND 0,E,10
,X:GOSUB 100:NEXT X
300 IF Z$="N" AND A(E-10)=47 THEN U=1
310 IF Z$="S" AND A(E+10)=47 THEN U=1
320 IF Z$="E" AND A(E+1)=47 THEN U=1
330 IF Z$="W" AND A(E-1)=47 THEN U=1
340 IF U=1 THEN ? :? "Blocked tunnel !!.
..":FOR X=0 TO 3:FOR Y=3 TO 28
350 IF U=1 THEN SOUND 0,Y,10,15:SOUND 1,
Y+3,10,15:NEXT Y:NEXT X:SOUND 0,0,0,0:SO
UND 1,0,0,0:GOSUB 110:GOTO 190
360 A(E)=46:E=E+(Z$="E")-(Z$="W")+(Z$="S
")*10-(Z$="N")*10:F=E
370 IF A(F)=63 THEN GOSUB 440
380 IF A(F)=66 THEN GOSUB 550
390 IF A(F)=81 THEN GOSUB 680
400 IF A(F)=71 THEN GOSUB 730
410 GOSUB 110:H=H+1:IF H=25 THEN Q=9:GOS
UB 980
420 GOTO 180
430 REM * *  SPACEWARP CAVE  * *
440 GOSUB 110:? "WARNING:You have just e
ntered a cave"
450 ? "containing a SPACEWARP!!.."
460 ? :? " You are now going to be trans
ported"
```

```
470 ? "to another cave....STAND BY!"
480 FOR X=4 TO 50 STEP 5:GOSUB 100
490 FOR Y=30 TO 250 STEP X
500 FOR Z=0 TO 3:SOUND Z,Y+Z,10,15:NEXT
Z:NEXT Y:NEXT X
510 FOR Z=1 TO 3:SOUND Z,0,0,0:NEXT Z
520 FOR X=255 TO 0 STEP -2:SOUND 0,X,12,
15:NEXT X:SOUND 0,0,0,0
530 A(E)=0:E=INT(RND(0)*76+12):RETURN
540 REM * *  BEASTIE CAVE  * *
550 GOSUB 110:? "WHOOPS! There's a BEAST
IE in here!!!"
560 FOR X=1 TO 15:FOR P=255 TO 50 STEP -
25:SOUND 0,P,10,15:NEXT P:GOSUB 100:NEXT
 X:SOUND 0,0,0,0
570 M=RND(0):IF M<0.2 THEN ? :? "But luc
kily for you , it has not":? "detected y
our presence":GOTO 640
580 ? "AND IT SEES YOU !!!......."
590 FOR X=255 TO 0 STEP -1:SOUND 0,X,10,
15:NEXT X
600 IF M>0.8 THEN ? :? "But it decides n
ot to eat you because":? "humans give hi
m indigestion.":GOTO 640
610 ? :? "AND HE EATS YOU....."
620 FOR P=1 TO 3:? "MUNCH..";:GOSUB 100:
FOR X=0 TO 5:FOR Y=1 TO 5:SOUND 0,X,8,X:
NEXT Y
630 NEXT X:NEXT P:SOUND 0,0,0,0:GOTO 980
640 ? :? "HIT ANY KEY":POKE 764,255
650 IF PEEK(764)<>255 THEN GET #1,A:POKE
 764,255:RETURN
660 FOR X=1 TO 30:NEXT X:GOSUB 100:GOTO
650
670 REM * * QUICKSAND CAVE * *
680 GOSUB 110:? "You have entered a cave
 containing"
690 ? "  QUICKSAND!!!!!!!......"
700 FOR P=1 TO 35:X=INT(RND(0)*8+4):SOUN
D 0,X,8,15:FOR Y=1 TO 30:NEXT Y:GOSUB 10
```

```
0:NEXT P
710 GOTO 980
720 REM * *  GANYMEDE GEMS CAVE  * *
730 GOSUB 110:? " Well done , you have f
ound some "
740 ? "          Ganymede gems!!"
750 ? :? "         They are worth $";
760 K=INT(RND(0)*100)+50:? K;" !"
770 G=G+K:FOR Y=1 TO 20:FOR X=3+Y*3 TO 2
0+Y*3:SOUND 0,X,10,15:NEXT X:GOSUB 100:N
EXT Y:SOUND 0,0,0,0
780 RETURN
790 REM * *  SECTOR CAVE SCAN  * *
800 L=46:IF A(E-11)<>46 THEN L=A(E-11)
810 IF A(E-10)<>46 THEN L=A(E-10)
820 IF A(E-9)<>46 THEN L=A(E-9)
830 IF A(E-1)<>46 THEN L=A(E-1)
840 IF A(E+1)<>46 THEN L=A(E+1)
850 IF A(E+9)<>46 THEN L=A(E+9)
860 IF A(E+10)<>46 THEN L=A(E+10)
870 IF A(E+11)<>46 THEN L=A(E+11)
880 IF L=46 THEN RETURN
890 ? "Computer reports:-"
900 IF L=47 THEN ? "Blocked tunnel ";
910 IF L=63 THEN ? "Space warp ";
920 IF L=66 THEN ? "Ganymede BEASTIE ";
930 IF L=81 THEN ? "Quicksand cave ";
940 IF L=71 THEN ? "Ganymede gems ";
950 ? "nearby":? :?
960 RETURN
970 REM * *  END OF MISSION  * *
980 IF Q<>0 THEN 1000
990 ? :? "END OF MISSION - YOU ARE DEAD"
1000 A(E)=72:? :? "YOU SURVIVED FOR ";H;
" MINUTES";
1010 IF H=25 THEN ? " ..WELL DONE"
1020 IF H<>25 THEN PRINT
1030 ? "AND FOUND GEMS WORTH $";G
```

```
1040 ? :? "Ganymede sector profile:":?
1050 FOR J=0 TO 90 STEP 10
1060 M=A(J+1):N=A(J+2):O=A(J+3):P=A(J+4)
:Z=A(J+5):R=A(J+6):S=A(J+7):T=A(J+8):U=A
(J+9):V=A(J+10)
1070 ? CHR$(M);CHR$(N);CHR$(O);CHR$(P);C
HR$(Z);CHR$(R);CHR$(S);CHR$(T);CHR$(U);C
HR$(V):NEXT J
1080 FOR J=1 TO 200:SOUND 0,J,4,10:NEXT
J:SOUND 0,0,0,0
1090 IF Q<>0 THEN END
1100 ? "HIT START KEY TO CONTINUE"
1110 POKE 53279,0:IF PEEK(53279)<>6 THEN
 1110
1120 GOSUB 110:RETURN
```

MAZE-RUNNER

If you like difficult puzzles, then you'll love this game. It took me more than a week to design the maze, and longer than that to convert it into numbers which the computer understands.

The maze is three times bigger than the screen size of a graphics mode 5 screen!

I designed this program to utilize a method of scrolling which I thought of. By altering the fourth and fifth byte of the display list (which are pointers to the screen memory area), you can scroll up or down by adding or subtracting the number of bytes in a scan line. This is the technique used in this program.

Once you've given up trying to solve the maze, you can see 'Arnold Atari' do it.

```
1 REM ****************************
2 REM * MAZE RUNNER - - - WRITTEN  *
3 REM * BY PAUL BUNN AND IAN NICOL *
4 REM ****************************
5 REM
6 REM
10 FOR J=1536 TO 1536+7:READ C:POKE J,C:
NEXT J:DIM G$(1)
20 DATA 104,165,203,37,204,133,205,96
30 GRAPHICS 5:SETCOLOR 0,4,6
40 A=PEEK(560)+256*PEEK(561):POKE 752,1
50 W=PEEK(A+4)+256*PEEK(A+5)
60 W=W-2500:X=INT(W/256):Y=INT(W-X*256):
B=W
70 POKE A+4,Y:POKE A+5,X:IF PEEK(W)=85 T
HEN 110
80 ? "PLEASE WAIT.":TRAP 110:RESTORE 770
90 READ C
100 POKE B,C:B=B+1:GOTO 90
110 TRAP 40000:RESTORE 110
120 REM ** MAIN LOOP **
130 ? "Would you like arnold to try it "
;:INPUT G$:IF G$="Y" THEN E=1:Y=E
140 FOR I=W+2319 TO W+2319+(3*20) STEP 2
0:POKE I,255:NEXT I
150 L=480+W:M=16:? "}":POKE 752,1
160 POKE L,(PEEK(L)+M)
170 IF  NOT E THEN O=STICK(0):POKE 77,0
180 R=R-1
190 IF E AND R<1 THEN READ O,R
200 IF O<>15 THEN POKE 53279,0
210 IF PEEK(L+1)=255 AND M=1 THEN GOTO 7
00
220 IF O=14 THEN GOSUB 420:POKE L,(PEEK(
L)-M):L=L-20:GOSUB 270:GOTO 160
230 IF O=13 THEN GOSUB 450:POKE L,(PEEK(
L)-M):L=L+20:GOSUB 300:GOTO 160
240 IF O=7 THEN POKE L,PEEK(L)-M:M=INT(M
/4):GOSUB 370:GOTO 160
```

```
250 IF O=11 THEN POKE L,PEEK(L)-M:M=M*4:
GOSUB 320:GOTO 160
260 GOTO 170
270 P=PEEK(A+4):P=P-20
280 IF P<0 THEN POKE A+5,PEEK(A+5)-1:P=P
+256
290 POKE A+4,P:RETURN
300 P=PEEK(A+4):P=P+20:IF P>255 THEN POK
E A+5,PEEK(A+5)+1:P=P-256
310 POKE A+4,P:RETURN
320 REM
330 IF M>255 THEN M=1:L=L-1
340 POKE 203,PEEK(L):POKE 204,M:K=USR(15
36)
350 IF PEEK(205)=M THEN M=INT(M/4):GOSUB
 370:GOSUB 480
360 RETURN
370 REM
380 IF M=0 THEN M=64:L=L+1
390 POKE 203,PEEK(L):POKE 204,M:K=USR(15
36)
400 IF PEEK(205)=M THEN M=M*4:GOSUB 320:
GOSUB 480
410 RETURN
420 J=PEEK(L-20):POKE 203,J:POKE 204,M:K
=USR(1536)
430 IF PEEK(205)=M THEN POP :GOSUB 480:G
OTO 170
440 RETURN
450 J=PEEK(L+20):POKE 203,J:POKE 204,M:K
=USR(1536)
460 IF PEEK(205)=M THEN POP :GOSUB 480:G
OTO 170
470 RETURN
480 FOR I=10 TO 0 STEP -1:SOUND 0,(L/256
),10,I:NEXT I:RETURN
490 DATA 14,10,7,4,14,4,11,4,14,9
500 DATA 7,4,13,4
510 DATA 7,6,13,3,7,2,14,2,7,2,13,5,11,1
,13,2,7,2,13,4,7,5,13,3
```

```
520 DATA 7,12,14,10
530 DATA 11,7,14,2,7,11,14,3,7,2,13,5,11
,4,13,10,7,1,13,2,11,2
540 DATA 13,5,7,4,14,9,11,1,14,2,7,3,13,
13,8,100,11,4,13,5,7,2,13,6,7,3,8,100
550 DATA 13,2,7,9,13,31,11,21
560 DATA 14,8,11,6,14,2,7,8,13,8,7,17,14
,2,11,3,14,2,7,3,14,2,11,3,14,2,11,6,14,
4,7,2,13,2,7,7
570 DATA 14,2,11,5,14,2,7,5,14,2,11,5,14
,2,7,5,14,9,11,6,13,2,8,75
580 DATA 7,4,13,5,11,5,13,2,11,4,14,2,7,
1,14,5,11,2,13,3
590 DATA 11,2,13,1,11,5,14,2,7,3,14,1,7,
2,14,9,11,2,14,16,11,3,13,2,11,2,13,2,7,
3,13,2,11,3,13,2,7,3
600 DATA 13,2,11,3,13,2,7,3,13,6
610 DATA 7,2,13,2,11,2,13,4,11,2,14,10,1
1,5,13,2,7,3,13,2,11,3,13,2,7,3,13,2,11,
3,13,2,7,3,13,2,7,1,13,2,11,3,14,1
620 DATA 11,3,13,2,11,3,13,3,7,2,14,1,7,
2,13,3,11,1,13,2,7,2,13,2,11,5,13,2,7,3,
13,6,11,2,14,1,11,5,14,1
630 DATA 11,6,13,3,11,3,13,3,11,4,13,2,7
,6,14,3,7,3,14,3,7,2,13,1,7,4,13,5,11,2,
14,3,11,2,13,2,11,3,13,21,7,4
640 DATA 14,10,7,4,13,11,7,4,13,2,7,23,1
3,15,11,30,13,4,7,11,13,2,7,2,14,4,7,2,1
3,4,7,2,14,4,7,2,13,4,7,8
650 DATA 14,2,11,6,14,2,7,8,13,4,7,2,14,
4,7,1,14,1,7,6,8,200
660 DATA 14,4,7,4,14,2,7,2,14,16,11,2,14
,9,7,18,13,2,11,16,13,5,7,2,13,2,7,2,14,
5,7,4,13,2,11,2,13,5,8,250
670 DATA 11,4,13,4,7,4,13,2,11,4,13,5,7,
5,14,2,7,2,14,1,7,2,13,1,7,2,14,1,7,2,13
,1,7,1,13,2,11,4
680 DATA 13,2,7,4,13,2,11,6,14,2,11,4,13
,2,7,2,13,2,11,2,13,3,11,2,13,3,7,2,14,1
,7,2,14,3
690 DATA 7,2,14,2,7,6,13,2,11,4,13,2,11,
2,13,2,7,4,14,2,7,2,13,2,7,7
```

```
700 REM ****** YOU WIN *******
710 FOR J=15 TO 0 STEP -2:FOR W=100 TO 2
00 STEP 12:SOUND 0,(J+10)*10,10,12
720 SOUND 1,W,10,12:SETCOLOR 0,(RND(0)*1
6),8:NEXT W:NEXT J
730 SETCOLOR 0,6,4:SOUND 0,0,0,0:SOUND 1
,0,0,0
740 GRAPHICS 0:POKE 752,1:? "      Well d
one , you managed to escape"
750 ? "the death-defying maze.You are on
e of":? "great intelligence to complete
such"
760 ? "an incredible task."
770 DATA 85,85,85,85,85,85,85,85,85,85
780 DATA 85,85,85,85,85,85,85,85,85,85
790 DATA 64,4,4,68,4,0,0,0,0,68
800 DATA 0,64,0,16,0,0,0,0,0,1
810 DATA 64,4,68,4,68,85,85,85,84,4
820 DATA 84,69,85,17,85,85,85,85,85,81
830 DATA 65,4,4,68,68,64,0,0,0,68
840 DATA 4,64,1,17,0,64,64,64,0,17
850 DATA 65,5,20,64,68,69,85,85,85,85
860 DATA 68,85,81,17,20,4,4,84,85,17
870 DATA 65,0,0,85,64,64,1,1,1,64
880 DATA 68,64,17,1,5,85,68,4,65,17
890 DATA 65,68,68,64,85,85,16,16,16,68
900 DATA 68,85,17,85,64,0,85,68,5,17
910 DATA 69,69,84,68,64,16,21,85,85,68
920 DATA 68,64,16,0,85,84,0,69,65,17
930 DATA 69,0,68,4,69,17,80,0,0,4
940 DATA 68,69,85,84,0,5,84,64,81,17
950 DATA 69,84,69,20,65,16,17,85,85,84
960 DATA 68,64,0,5,85,64,4,84,81,17
970 DATA 64,4,69,20,81,21,16,0,4,0
980 DATA 68,85,85,64,0,85,68,4,1,17
990 DATA 85,68,5,16,65,0,21,85,68,85
1000 DATA 68,0,0,85,84,64,69,69,81,17
1010 DATA 85,68,84,17,81,85,81,0,68,64
1020 DATA 5,85,84,64,0,84,64,64,65,17
```

```
1030 DATA 69,68,0,80,17,1,1,17,68,69
1040 DATA 85,0,4,68,85,84,84,84,69,17
1050 DATA 64,4,21,81,17,17,17,16,68,64
1060 DATA 1,21,68,68,64,0,4,4,68,17
1070 DATA 69,84,16,1,16,16,17,20,68,69
1080 DATA 81,16,4,68,69,85,69,68,68,17
1090 DATA 68,0,17,85,21,85,81,16,68,64
1100 DATA 17,17,84,68,68,0,64,68,68,17
1110 DATA 68,85,81,1,0,5,81,17,68,69
1120 DATA 17,16,4,69,68,84,84,68,69,17
1130 DATA 68,80,1,21,85,68,0,16,68,65
1140 DATA 16,21,68,64,4,4,4,68,65,17
1150 DATA 68,85,21,16,0,69,85,85,68,81
1160 DATA 21,80,4,85,85,69,68,68,85,17
1170 DATA 68,1,16,17,84,64,0,0,4,17
1180 DATA 21,81,84,0,0,0,68,68,1,17
1190 DATA 69,1,17,16,68,69,84,85,85,17
1200 DATA 21,80,5,85,85,84,68,68,21,17
1210 DATA 65,5,17,20,64,68,0,64,16,17
1220 DATA 21,85,64,0,0,68,68,69,80,17
1230 DATA 65,68,0,4,85,68,85,69,17,81
1240 DATA 16,0,85,85,84,68,68,64,17,17
1250 DATA 64,69,85,68,0,68,4,1,17,1
1260 DATA 21,84,0,0,0,68,68,85,17,17
1270 DATA 84,64,0,69,84,69,68,85,17,17
1280 DATA 16,21,85,85,85,68,68,64,17,17
1290 DATA 84,85,84,64,0,64,68,1,17,81
1300 DATA 17,0,64,0,0,68,68,69,81,17
1310 DATA 84,0,4,85,69,68,69,81,16,1
1320 DATA 17,84,69,85,84,68,64,64,81,17
1330 DATA 64,21,68,0,4,4,68,1,17,85
1340 DATA 17,0,68,0,64,68,85,84,65,17
1350 DATA 68,85,69,68,84,84,68,85,17,0
1360 DATA 17,21,68,68,4,0,4,4,69,17
1370 DATA 68,85,64,68,64,64,68,1,17,21
1380 DATA 81,4,4,69,85,85,68,68,68,17
1390 DATA 68,5,64,68,69,69,69,81,17,16
1400 DATA 1,64,68,65,0,0,68,68,68,81
```

```
1410 DATA 69,85,68,68,68,4,4,1,17,16
1420 DATA 21,85,85,85,17,84,68,68,4,65
1430 DATA 64,0,4,68,68,84,84,85,17,21
1440 DATA 0,0,1,85,17,4,68,69,84,85
1450 DATA 69,21,84,68,69,64,64,1,17,1
1460 DATA 85,85,80,1,17,68,68,64,0,1
1470 DATA 68,16,0,68,64,69,85,81,17,81
1480 DATA 0,0,21,81,81,4,68,69,85,85
1490 DATA 64,80,85,68,68,68,0,17,16,17
1500 DATA 21,85,16,17,17,20,68,68,4,5
1510 DATA 68,65,64,4,68,68,85,17,21,17
1520 DATA 4,5,17,17,17,4,68,68,68,69
1530 DATA 68,65,5,84,68,68,1,17,1,17
1540 DATA 64,65,17,17,17,68,68,68,68,69
1550 DATA 68,65,20,0,68,69,81,17,81,17
1560 DATA 85,85,17,17,16,4,68,68,68,69
1570 DATA 68,64,16,85,68,68,1,16,17,16
1580 DATA 0,0,17,17,17,68,68,64,64,65
1590 DATA 68,85,81,64,68,68,85,21,17,17
1600 DATA 69,85,81,17,17,68,68,85,85,81
1610 DATA 64,0,1,5,68,68,1,16,17,17
1620 DATA 0,0,1,1,17,4,68,0,0,21
1630 DATA 81,84,85,20,4,69,81,17,81,21
1640 DATA 21,85,81,85,17,20,69,85,85,1
1650 DATA 65,84,81,16,84,68,1,16,17,0
1660 DATA 16,0,17,84,1,16,64,0,1,81
1670 DATA 69,4,1,16,64,68,85,17,81,85
1680 DATA 81,85,17,84,85,17,81,85,80,1
1690 DATA 68,4,69,17,69,68,1,1,81,1
1700 DATA 16,1,17,84,64,17,17,16,21,85
1710 DATA 68,68,68,16,4,5,81,85,1,17
1720 DATA 1,81,17,84,69,81,17,17,0,1
1730 DATA 68,68,68,81,84,84,80,64,21,17
1740 DATA 21,81,17,84,69,17,17,17,85,81
1750 DATA 68,68,64,65,0,64,20,69,80,17
1760 DATA 0,1,17,80,69,17,17,1,0,1
1770 DATA 64,64,68,69,21,69,0,64,1,81
1780 DATA 85,81,17,81,69,17,17,85,21,85
```

```
1790 DATA 85,85,68,68,16,5,81,85,85,65
1800 DATA 64,1,17,65,5,17,16,0,0,1
1810 DATA 64,0,4,68,81,84,17,0,0,69
1820 DATA 69,85,17,69,20,17,21,85,85,65
1830 DATA 85,21,84,68,65,0,85,21,84,64
1840 DATA 4,0,17,4,16,17,0,0,0,65
1850 DATA 64,16,0,68,64,17,64,0,4,69
1860 DATA 84,85,81,20,81,17,85,85,84,65
1870 DATA 69,81,85,68,85,80,69,85,68,84
1880 DATA 4,0,17,16,65,16,0,5,84,85
1890 DATA 64,1,0,4,0,64,68,0,4,4
1900 DATA 85,85,17,17,69,21,69,64,0,1
1910 DATA 85,85,21,85,84,69,68,85,85,68
1920 DATA 0,0,17,17,85,0,68,85,85,85
1930 DATA 64,0,21,1,0,64,68,64,64,69
1940 DATA 84,85,81,16,0,4,4,64,1,1
1950 DATA 85,20,80,17,21,84,68,4,4,68
1960 DATA 4,0,17,21,85,85,84,85,81,17
1970 DATA 64,20,1,81,16,0,69,85,84,68
1980 DATA 69,85,17,0,0,0,16,0,81,17
1990 DATA 69,85,85,1,17,84,64,0,4,68
2000 DATA 64,0,17,85,81,81,17,84,81,17
2010 DATA 68,0,0,21,16,5,85,85,68,68
2020 DATA 85,85,81,0,16,17,16,4,17,17
2030 DATA 68,85,85,81,21,68,0,0,68,68
2040 DATA 0,0,17,21,21,17,17,69,17,17
2050 DATA 68,0,0,1,21,68,85,84,68,68
2060 DATA 85,69,81,17,1,17,16,65,17,17
2070 DATA 68,1,85,85,0,68,0,4,68,68
2080 DATA 64,64,17,17,81,17,21,81,17,17
2090 DATA 69,81,0,1,85,69,85,68,68,68
2100 DATA 84,85,17,17,81,17,16,17,1,17
2110 DATA 64,1,21,80,0,68,0,68,68,68
2120 DATA 4,64,17,16,17,17,17,17,85,17
2130 DATA 84,85,16,21,84,0,68,68,68,69
2140 DATA 68,69,81,17,81,17,17,16,0,17
2150 DATA 84,64,17,0,21,85,68,68,68,64
2160 DATA 68,64,17,16,17,17,17,21,85,81
```

```
2170 DATA 64,69,81,85,16,0,4,68,68,85
2180 DATA 84,85,17,21,17,17,1,0,16,1
2190 DATA 85,68,1,1,17,85,84,68,0,0
2200 DATA 0,0,17,16,1,17,85,81,17,85
2210 DATA 64,4,85,17,16,0,4,69,85,85
2220 DATA 85,85,81,21,81,17,16,17,16,1
2230 DATA 69,84,64,17,21,85,68,64,0,0
2240 DATA 0,0,1,0,17,17,17,17,21,81
2250 DATA 64,0,69,80,16,0,4,85,85,85
2260 DATA 85,85,85,85,17,17,17,17,16,1
2270 DATA 85,85,68,21,81,85,84,0,0,0
2280 DATA 0,0,0,1,21,17,17,1,17,85
2290 DATA 64,64,68,4,16,0,5,85,85,85
2300 DATA 85,85,85,81,0,17,17,85,16,1
2310 DATA 68,68,68,84,85,85,64,0,0,0
2320 DATA 0,0,0,17,85,81,17,1,21,81
2330 DATA 68,4,4,68,64,0,85,85,85,85
2340 DATA 85,85,85,16,0,1,0,17,0,1
2350 DATA 85,85,68,68,69,84,64,64,0,0
2360 DATA 0,0,1,21,85,85,85,85,85,81
2370 DATA 64,0,68,68,68,0,68,69,85,85
2380 DATA 85,85,81,0,0,0,0,0,0,17
2390 DATA 69,20,68,68,69,69,68,68,0,0
2400 DATA 0,0,17,85,85,85,85,85,85,17
2410 DATA 69,16,68,68,64,68,64,68,85,85
2420 DATA 85,85,17,1,0,0,0,0,1,17
2430 DATA 69,17,68,68,84,68,81,68,64,0
2440 DATA 64,1,17,81,21,85,85,85,81,17
2450 DATA 69,16,68,64,4,64,0,4,69,84
2460 DATA 69,81,1,1,16,0,0,0,1,17
2470 DATA 69,20,68,68,68,85,85,84,68,0
2480 DATA 68,81,85,21,17,85,85,85,85,17
2490 DATA 69,16,68,68,68,0,16,64,68,85
2500 DATA 68,17,1,1,17,16,1,0,1,17
2510 DATA 69,17,68,68,69,85,16,69,68,64
2520 DATA 5,17,17,81,17,1,81,21,81,17
2530 DATA 69,16,68,68,68,0,16,68,4,69
2540 DATA 84,16,16,1,17,81,1,16,1,17
```

```
2550 DATA 69,20,68,68,68,85,84,68,84,68
2560 DATA 0,85,85,85,16,17,21,17,85,17
2570 DATA 69,16,68,68,68,68,0,4,64,68
2580 DATA 85,80,16,21,21,17,16,16,1,17
2590 DATA 69,17,68,68,68,64,69,84,69,68
2600 DATA 64,1,17,1,1,1,17,85,81,17
2610 DATA 69,16,68,68,68,85,65,64,68,4
2620 DATA 85,85,0,21,81,85,17,0,17,17
2630 DATA 69,21,69,68,68,0,64,68,68,68
2640 DATA 0,0,21,16,17,0,1,21,17,17
2650 DATA 69,0,0,4,69,84,84,68,69,69
2660 DATA 85,85,80,17,17,21,85,17,17,17
2670 DATA 85,21,69,84,64,4,4,4,64,64
2680 DATA 0,1,1,81,17,16,1,17,1,17
2690 DATA 69,16,69,84,85,69,69,85,85,85
2700 DATA 85,81,21,1,17,21,81,17,85,17
2710 DATA 68,17,68,0,0,64,0,0,0,0
2720 DATA 0,17,16,21,17,0,17,0,1,17
2730 DATA 68,81,69,85,84,85,85,85,85,81
2740 DATA 85,17,17,81,17,85,17,85,81,17
2750 DATA 68,64,68,0,0,64,0,0,0,1
2760 DATA 1,17,17,1,17,0,16,0,1,17
2770 DATA 68,68,68,85,85,69,85,85,85,85
2780 DATA 17,17,17,21,17,21,85,85,85,17
2790 DATA 68,68,68,1,1,4,0,0,68,5
2800 DATA 17,17,17,1,17,16,20,4,5,17
2810 DATA 68,68,69,81,17,20,85,85,68,69
2820 DATA 17,17,17,81,17,17,0,64,65,17
2830 DATA 64,68,64,17,17,20,85,80,4,69
2840 DATA 17,1,0,1,17,21,69,85,81,17
2850 DATA 69,68,85,17,17,20,0,1,68,69
2860 DATA 17,21,85,21,17,0,5,16,1,17
2870 DATA 64,4,1,17,17,21,85,80,68,69
2880 DATA 16,16,0,16,17,85,85,17,85,17
2890 DATA 85,84,85,17,17,0,0,20,64,65
2900 DATA 21,17,85,81,81,1,0,16,1,17
2910 DATA 64,0,1,17,17,85,85,20,85,85
2920 DATA 1,16,16,1,1,17,21,21,81,17
```

```
2930 DATA 69,84,85,17,17,1,1,0,0,1
2940 DATA 81,21,17,85,21,17,1,0,1,17
2950 DATA 64,20,81,17,17,17,17,85,85,81
2960 DATA 16,16,17,0,16,17,81,85,85,17
2970 DATA 85,16,65,16,16,16,16,0,0,1
2980 DATA 1,17,81,21,81,81,1,0,1,17
2990 DATA 64,17,69,21,85,85,85,85,85,85
3000 DATA 85,16,1,16,1,1,21,21,81,17
3010 DATA 69,81,1,16,0,0,0,0,0,0
3020 DATA 0,21,85,21,21,21,16,16,1,17
3030 DATA 64,17,81,17,85,85,85,85,85,85
3040 DATA 85,80,0,1,16,16,17,81,85,17
3050 DATA 85,17,1,17,0,0,0,4,4,0
3060 DATA 0,65,85,81,17,81,81,1,1,85
3070 DATA 64,17,21,17,85,85,84,68,68,85
3080 DATA 84,69,0,1,17,17,1,21,16,1
3090 DATA 69,81,1,16,0,0,4,68,68,0
3100 DATA 4,68,21,85,17,0,17,0,16,1
3110 DATA 64,17,81,21,85,85,68,68,69,85
3120 DATA 68,68,80,0,17,85,85,85,84,1
3130 DATA 85,16,1,0,0,0,64,64,64,0
3140 DATA 4,4,1,85,16,0,0,0,4,1
3150 DATA 85,85,85,85,85,85,85,85,85,85
3160 DATA 85,85,85,85,85,85,85,85,85,85
```

ASTRO BLAST

Our next game was written by Paul Dunning. The game is for two players, each controlling his or her own ship at the same time. You control your ship by using the joystick in port one for player one and in port two for player two. Pushing the joystick up will give you thrust, and move you one square in the direction you are facing. Push the joystick left or right to rotate the ship.

When you first run the program, the computer will ask the time to play. About 30 is average. The object of the game is to hit your opponent as many times as possible in the allocated time. Note that you have a limited number of shots, with the number of remaining energy shells shown at the bottom of the screen. Now, when you're ready, blast away!

```
1 REM *******************************
2 REM *         Astro blast         *
3 REM *         -----------         *
4 REM *   Written by Paul Dunning   *
5 REM *******************************
6 REM
10 DIM F1$(20),F2$(20),X(8),Y(8):F1$(1)=
CHR$(172):F1$(20)=CHR$(172):F1$(2)=F1$:F
2$=",,,,,,,,,,,,,,,"
20 RAM=PEEK(106)-16:POKE 756,RAM:IF PEEK
(203)<>200 THEN GOSUB 890:POKE 203,200
30 RESTORE 960:FOR Q=1 TO 8:READ XX,YY:X
(Q)=XX:Y(Q)=YY:NEXT Q:C1=1:C=1
40 GRAPHICS 2
50 ? #6;"      astro blast "
60 ? #6;"     **************"
70 ? :? " ENTER TIME TO PLAY ";:INPUT TI
80 REM
90 GRAPHICS 17:POKE 756,RAM:F1=14:F2=14:
X=5:Y=14:Q=15:W=5:FF1=1:FF2=1
100 S=STICK(0):IF S=7 THEN C=C+1:IF C>8
THEN C=1
110 IF S=11 THEN C=C-1:IF C<1 THEN C=8
120 IF S=14 THEN COLOR 32:PLOT X,Y:X=X+X
(C):Y=Y+Y(C):SOUND 0,200,8,8
130 IF S<>14 THEN SOUND 0,0,0,0
140 IF X<2 OR X>18 THEN X=X+-X(C):X=(X=2
)*18+(X=18)*2
150 IF Y<2 OR Y>18 THEN Y=Y+-Y(C):Y=(Y=2
)*18+(Y=18)*2
160 COLOR C:PLOT X,Y
170 IF STRIG(0)=0 AND FF1=1 THEN GOSUB 3
40
180 IF STRIG(1)=0 AND FF2=1 THEN GOSUB 5
20
190 S=STICK(1):IF S=7 THEN C1=C1+1:IF C1
>8 THEN C1=1
200 IF S=11 THEN C1=C1-1:IF C1<2 THEN C1
=8
210 IF S=14 THEN COLOR 32:PLOT Q,W:Q=Q+X
```

```
(C1):W=W+Y(C1):SOUND 1,100,8,8
220 IF S<>14 THEN SOUND 1,0,0,0
230 IF Q<2 OR Q>18 THEN Q=Q+-X(C1):Q=(Q=
2)*18+(Q=18)*2
240 IF W<2 OR W>18 THEN W=W+-Y(C1):W=(W=
18)*2+(W=2)*18
250 COLOR C1:PLOT Q,W
260 IF F1<1 THEN POSITION 0,21:? #6;"out
 of fuel":FF1=0:GOTO 280
270 POSITION 0,21:? #6;"FUEL1";F1$(1,F1)
;"  "
280 IF F2<1 THEN POSITION 0,22:? #6;"out
 of fuel":FF2=0:GOTO 300
290 POSITION 0,22:? #6;"FUEL2";F2$(1,F2)
;"  "
300 POSITION 5,0:? #6;S1:POSITION 15,0:?
 #6;S2:POSITION 8,0:? #6;"TIME":POSITION
 9,1:? #6;INT(TI);" "
310 IF PEEK(53279)=6 THEN GOTO 90
320 TI=TI-0.1:IF TI<0.1 THEN 800
330 GOTO 100
340 X1=X+X(C):Y1=Y+Y(C)
350 IF C=1 OR C=5 THEN FC=10
360 IF C=3 OR C=7 THEN FC=12
370 IF C=2 OR C=6 THEN FC=11
380 IF C=4 OR C=8 THEN FC=13
390 FOR F=1 TO 5:LOCATE X1,Y1,XX:COLOR F
C:PLOT X1,Y1
400 IF XX<>32 AND XX<10 THEN GOSUB 700:S
1=S1+1:HH=1:GOTO 450
410 SOUND 0,255-F*50,8,15:SOUND 1,150-F*
30,8,155
420 IF X1>18 OR X1<1 THEN 450
430 IF Y1>18 OR Y1<1 THEN 450
440 X1=X1+X(C):Y1=Y1+Y(C):NEXT F
450 X1=X+X(C):Y1=Y+Y(C)
460 FOR E=1 TO 5:COLOR 32:PLOT X1,Y1
470 IF X1>18 OR X1<1 THEN 500
480 IF Y1>19 OR Y1<1 THEN 500
490 X1=X1+X(C):Y1=Y1+Y(C):NEXT E
```

```
500 SOUND 0,0,0,0:SOUND 1,0,0,0:F1=F1-1:
IF HH=1 THEN HH=0:GOTO 90
510 RETURN
520 X2=Q+X(C1):Y2=W+Y(C1)
530 IF C1=1 OR C1=5 THEN FC=10
540 IF C1=3 OR C1=7 THEN FC=12
550 IF C1=2 OR C1=6 THEN FC=11
560 IF C1=4 OR C1=8 THEN FC=13
570 FOR F=1 TO 5:LOCATE X2,Y2,XX:COLOR F
C:PLOT X2,Y2
580 SOUND 0,255-F*50,8,15:SOUND 1,150-F*
30,8,15
590 IF XX<>32 AND XX<10 THEN GOSUB 750:S
2=S2+1:HH=1:GOTO 630
600 IF X2>18 OR X2<1 THEN 630
610 IF Y2>18 OR Y2<1 THEN 630
620 X2=X2+X(C1):Y2=Y2+Y(C1):NEXT F
630 X2=Q+X(C1):Y2=W+Y(C1)
640 FOR E=1 TO 5:COLOR 32:PLOT X2,Y2
650 IF X2>18 OR X2<1 THEN 680
660 IF Y2>19 OR Y2<1 THEN 680
670 X2=X2+X(C1):Y2=Y2+Y(C1):NEXT E
680 SOUND 0,0,0,0:SOUND 1,0,0,0:F2=F2-1:
IF HH=1 THEN HH=0:GOTO 90
690 RETURN
700 FOR G=15 TO 0 STEP -3
710 FOR H=0 TO 2
720 SOUND H,RND(1)*50+100,8,G
730 NEXT H
740 NEXT G:FOR G=0 TO 2:SOUND G,0,0,0:NE
XT G:RETURN
750 FOR G=15 TO 0 STEP -3
760 FOR H=0 TO 2
770 SOUND H,RND(1)*50+100,8,G
780 NEXT H
790 NEXT G:FOR G=0 TO 2:SOUND G,0,0,0:NE
XT G:RETURN
800 POSITION 5,10:? #6;"GAME OVER"
810 S1=0:S2=0:FOR Q=1 TO 5
```

```
820 SOUND 0,100+C,10,15:SOUND 1,120+C,10
,15:SOUND 2,90+C,10,15
830 C=C-1:IF C=-30 THEN 850
840 GOTO 820
850 C=0:NEXT Q
860 FOR Q=0 TO 2:SOUND Q,0,0,0:NEXT Q
870 IF PEEK(53279)=6 THEN 40
880 POKE 53279,0:GOTO 870
890 FOR Q=0 TO 1024:A=PEEK(57344+Q):POKE
 RAM*256+Q,A:NEXT Q:C=8
900 READ A:IF A=-1 THEN RETURN
910 POKE RAM*256+C,A:C=C+1:GOTO 900
920 DATA 24,24,36,36,66,90,165,195,3,13,
50,194,52,20,8,8,192,176,76,35,35,76,176
,192,8,8,20,52,194,50,13,3
930 DATA 195,173,90,66,36,36,24,24,16,16
,40,44,67,76,176,192,3,13,50,196,196,50,
13,3,192,176,36,67,44,40,16,16
940 DATA 112,120,248,254,255,127,60,56,2
4,24,24,24,24,24,24,24,1,2,4,8,16,32,64,
128,0,0,0,255,255,0,0,0
950 DATA 128,64,32,16,8,4,2,1,-1
960 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0
,-1,-1
```

FROG JUMP

In FROG JUMP, another game from Paul Dunning, you control a frog using the joystick. The object of the game is to get five frogs safely home, past the road and river. You must dodge the cars on the road. You must jump onto passing boats on the river, but do *not* jump on the yellow boats.

Sometimes you may see your girlfriend on one of the boats. You can then get 200 bonus points by picking her up and taking her home. Chivalry brings its own reward.

If you manage to get all your frogs home, you'll receive a bonus based on the amount of time you have left. Then you'll advance to an even harder round.

```
10 REM *******************************
20 REM *         FROG - JUMP         *
30 REM *         -----------         *
40 REM *   WRITTEN BY PAUL DUNNING   *
50 REM *******************************
60 DIM P$(20),L1$(22),L2$(20),C1$(20),C2
$(20),T1$(20),T2$(20),X$(20),Y$(20),F$(4
):RAM=(PEEK(106)-16)*256
70 GRAPHICS 17
80 IF PEEK(RAM+9)=0 THEN GOSUB 1140
90 GRAPHICS 17:SETCOLOR 0,2,3:SETCOLOR 1
,12,5:SETCOLOR 2,13,8:SETCOLOR 3,2,6:SET
COLOR 4,8,1
100 X3=19:E=1:T1=50:J=1:M=3:X=10:Y=19:C=
7:POKE 756,RAM/256
110 P$(1)=CHR$(132):P$(20)=CHR$(132):P$(
2)=P$:F$(1)=CHR$(167):F$(4)=CHR$(167):F$
(2)=F$
120 L1$=" !!!!!     !!!!!    "
130 L2$="  !!  !!  !!  !!  !!"
140 FOR Q=1 TO 20:READ V:C1$(Q,Q)=CHR$(V
):NEXT Q
150 DATA 32,163,163,32,32,163,163,32,32,
163,163,32,32,163,163,32,32,163,163,32
160 C2$="  #   #   #   #   # "
170 FOR Q=1 TO 20:READ V:T1$(Q,Q)=CHR$(V
):NEXT Q
180 DATA 32,166,166,166,32,32,32,134,134
,134,32,32,32,134,134,134,32,32,32,32
190 FOR Q=1 TO 20:READ V:T2$(Q,Q)=CHR$(V
):NEXT Q
200 DATA 32,32,32,32,134,134,134,32,32,3
2,166,166,166,32,32,32,134,134,134,32
210 POSITION 0,10:? #6;P$
220 POSITION 0,20:? #6;P$
230 POSITION 0,0
240 FOR Q=1 TO 40:READ V:? #6;CHR$(V);:N
EXT Q
250 DATA 132,132,32,132,132,32,132,132,3
2,132,132,32,132,132,32,132,132,32,132,1
32
```

```
260 DATA 133,133,32,133,133,32,133,133,3
2,133,133,32,133,133,32,133,133,32,133,1
33
270 COLOR C:PLOT X,Y
280 POSITION 0,6:? #6;L1$:GOSUB 690:POSI
TION 0,8:? #6;T1$:GOSUB 690:POSITION 0,4
:? #6;T2$:POSITION 0,2:? #6;L2$
290 GOSUB 690:POSITION 0,12:? #6;C1$:GOS
UB 690:POSITION 0,15:? #6;C2$:GOSUB 690:
POSITION 0,18:? #6;C1$:GOSUB 690
300 COLOR OC:PLOT X,Y
310 LOCATE X,Y,CR
320 IF CR<>32 AND CR>10 AND Y>10 THEN GO
SUB 850
330 COLOR C:PLOT X,Y
340 X$=L2$(1,19):Y$=L2$(20,20):Y$(LEN(Y$
)+1)=X$:L2$=Y$
350 X$=L1$(1,19):Y$=L1$(20,20):Y$(LEN(Y$
)+1)=X$:L1$=Y$
360 X$=C1$(1,19):Y$=C1$(20,20):Y$(LEN(Y$
)+1)=X$:C1$=Y$
370 T1$=T1$(2,20):T1$(20,20)=T1$(1,1)
380 T2$=T2$(2,20):T2$(20,20)=T2$(1,1)
390 C2$=C2$(2,20):C2$(20,20)=C2$(1,1)
400 S=STICK(0):COLOR 2
410 POSITION 5,23:? #6;"score-";SC
420 POSITION 5,22:? #6;"time -";T;" "
430 POSITION 0,23:? #6;F$(1,M);" "
440 T1=T1-0.05:T=INT(T1)
450 IF S=7 THEN COLOR OC:PLOT X,Y:X=X+1:
LOCATE X,Y,OC:C=8:GOSUB 820
460 IF S=11 THEN COLOR OC:PLOT X,Y:X=X-1
:LOCATE X,Y,OC:C=9:GOSUB 820
470 IF S=14 THEN COLOR OC:PLOT X,Y:Y=Y-J
:LOCATE X,Y,OC:C=7:GOSUB 820
480 IF S=13 THEN COLOR OC:PLOT X,Y:Y=Y+J
:LOCATE X,Y,OC:C=10:GOSUB 820
490 IF OC<>32 AND OC>10 AND Y>10 THEN GO
SUB 850
500 IF OC=32 AND Y<10 AND Y>0 THEN GOSUB
 850
```

```
510 IF Y<11 THEN J=2
520 IF Y>10 THEN J=1
530 IF OC=134 THEN X=X-1
540 IF OC=7 THEN GOSUB 1320
550 IF OC=33 THEN X=X+1
560 IF OC=166 THEN X=X-1:TT=TT+1:IF TT=5
 THEN TT=0:X=X+1:GOSUB 850
570 IF Y=0 AND OC=32 THEN GOSUB 730:GOTO
 610
580 IF Y=0 THEN GOSUB 850
590 IF X>19 THEN X=X-1:GOSUB 850
600 IF X<0 THEN X=X+1:GOSUB 850
610 COLOR C:PLOT X,Y
620 IF RND(1)>0.95 AND HF=0 THEN HH=0
630 IF H=0 AND HH=0 T4EN GOSUB 1330:IF H
=0 THEN 670
640 IF HH=1 THEN 670
650 X3=X3-1:IF X3<2 THEN H=0:HF=1:HH=1
660 COLOR 7:PLOT X3,8
670 REM
680 GOTO 270
690 LOCATE X,Y,AC:IF AC<7 OR AC>10 THEN
OC=AC
700 IF H=1 THEN COLOR 7:PLOT X3,8
710 COLOR C:PLOT X,Y:RETURN
720 FOR W=1 TO 10:SOUND 0,W,10,15:NEXT W
730 X1=X:Y1=Y:X=10:Y=19
740 FOR W=1 TO 10:SOUND 0,W,10,15:NEXT W
750 FOR W=10 TO 1 STEP -1:SOUND 0,W,10,1
5:NEXT W
760 SOUND 0,0,0,0:COLOR C:PLOT X1,Y1
770 F=F+1:SC=SC+200:IF SC>2000 AND E=1 T
HEN E=0:M=M+1
780 HF=0
790 IF BB<>0 THEN SC=SC+BB:BB=0
800 IF F=6 THEN F=0:GOSUB 960:GOTO 210
810 RETURN
820 FOR W=55 TO 50 STEP -1
830 SOUND 0,W,10,15:NEXT W
```

```
840 SOUND 0,0,0,0:RETURN
850 FOR W=11 TO 15
860 COLOR W:PLOT X,Y:SOUND 0,W*8,10,15
870 FOR P=0 TO 50:NEXT P:NEXT W
880 COLOR 32:PLOT X,Y:M=M-1
890 COLOR OC:PLOT X,Y
900 SOUND 0,0,0,0:OC=32
910 IF M=0 THEN 930
920 X=10:Y=19:HF=0:RETURN
930 POSITION 5,10:? #6;"game over"
940 IF PEEK(53279)=6 THEN RUN
950 GOTO 940
960 GRAPHICS 17
970 IF T<10 THEN POSITION 5,10:? #6;"NO
BONUS":GOTO 1030
980 G=INT(RND(1)*400)+200
990 K=INT(T/10+LL):BO=K*G:SC=SC+BO
1000 POSITION 7,5:? #6;"BONUS"
1010 POSITION 2,7:? #6;K;" * ";G;" = ";B
O
1020 FOR W=1 TO 7
1030 FOR P=255 TO 0 STEP -13:POKE 708,P:
SOUND 0,P,10,10:SOUND 1,255-P,12,10:NEXT
 P
1040 NEXT W:SOUND 1,0,0,0
1050 GRAPHICS 17:SETCOLOR 0,2,3:SETCOLOR
 1,12,5:SETCOLOR 2,13,8:SETCOLOR 3,2,5:S
ETCOLOR 4,8,1:LL=LL+1
1060 IF LL>1 THEN 1110
1070 L1$=" !!!!!              "
1080 C2$="  ##  ##  ##  ##  ##"
1090 IF SC>2000 AND E=1 THEN E=0:M=M+1
1100 POKE 756,RAM/256:T1=50:RETURN
1110 L2$=" !!   !!   !!   !!  ":L2$(7,7)
=CHR$(186):L2$(8,8)=CHR$(187)
1120 GOTO 1090
1130 GOTO 270
1140 ? #6;" PLEASE WAIT -"
1150 ? #6;" REDEFING THE"
```

```
1160 ? #6;" CHARACTER SET"
1170 C=0:FOR Q=0 TO 2048
1180 A=PEEK(57344+Q)
1190 POKE RAM+Q,A
1200 NEXT Q
1210 POKE 756,RAM/256
1220 READ C:IF C=-1 THEN RETURN
1230 BA=RAM+(C*8)
1240 FOR Q=0 TO 7:READ V:POKE BA+Q,V:NEX
T Q
1250 GOTO 1220
1260 DATA 1,255,135,255,225,255,135,255,
255,3,0,119,34,254,255,254,34,119
1270 DATA 4,255,255,255,255,255,255,255,
255,5,255,255,255,255,255,255,255,0,6,13
0,214,124,124,124,124,214
1280 DATA 146,7,66,66,90,60,60,60,90,129
,8,128,71,56,124,124,56,71,128,9,1,226,2
8,62,62,28,226,1
1290 DATA 10,129,90,60,60,60,90,66,66,11
,73,42,20,42,20,42,73,0,12,0,42,20,42,20
,42,0,0
1300 DATA 13,0,8,20,42,20,8,0,0,14,0,0,2
0,8,20,0,0,0,15,0,60,66,90,90,66,60,0
1310 DATA 26,0,28,126,255,99,1,0,0,27,0,
6,15,6,142,254,248,112,-1
1320 FOR W=150 TO 50 STEP -1:SOUND 0,W,1
0,15:NEXT W:SOUND 0,0,0,0:BB=200:HF=1:H=
0:HH=1:X3=19:RETURN
1330 LOCATE X3,8,XX:IF XX<>32 AND XX<>16
6 THEN H=1
1340 X3=19
1350 RETURN
```

Chapter 14: USER-PROTECTING YOUR PROGRAMS

USER-PROTECTING YOUR PROGRAMS

Disabling the BREAK key:

This can be done using the following statement
POKE 16,64: POKE 53774,64
It must be done whenever the graphics mode is changed.

Using the cold start flag:

The command POKE 580,1 will cause the computer to do the power-up sequence when SYSTEM RESET is pressed. This means that if the user presses SYSTEM RESET, then the computer will clear all RAM, getting rid of the program.

The Variable name table:

This is where the computer stores the names of all the variables. Destroying the variable name table prevents the program from being listed. To try it, LOAD a program and then type the following lines in the direct mode:

```
FOR X= PEEK (130) + 256*PEEK (131) TO PEEK (132) +
256*PEEK (133): POKE X,0:NEXT X
```

Now try and list the program! When you save the program again, it will be saved in an unlistable form.

Here, in this small volume, in easy-to-understand form, Paul Bunn has brought together all the essential information you'll need to improve your programming techniques on the Atari computers.

From making the most of the graphics, to using sound and the joysticks or paddle, you'll find the vital addresses, the important locations, the programming tricks to get your Atari to do just about anything you want it to do.

And if you just want a generous collection of programs, in ready-to-run form, you'll also find them in this book. From BEETLE JUICE (you, as a small, red beetle, try to cross a busy street without being squashed) to DODGE 'EM (in which you use a joystick to drive your car around a maze, and avoid the computer's car), there are programs for every taste and occasion.

If you've come to the 'What do I do now?' stage with your Atari 400 or 800, then you're ready for Paul Bunn, and this book.