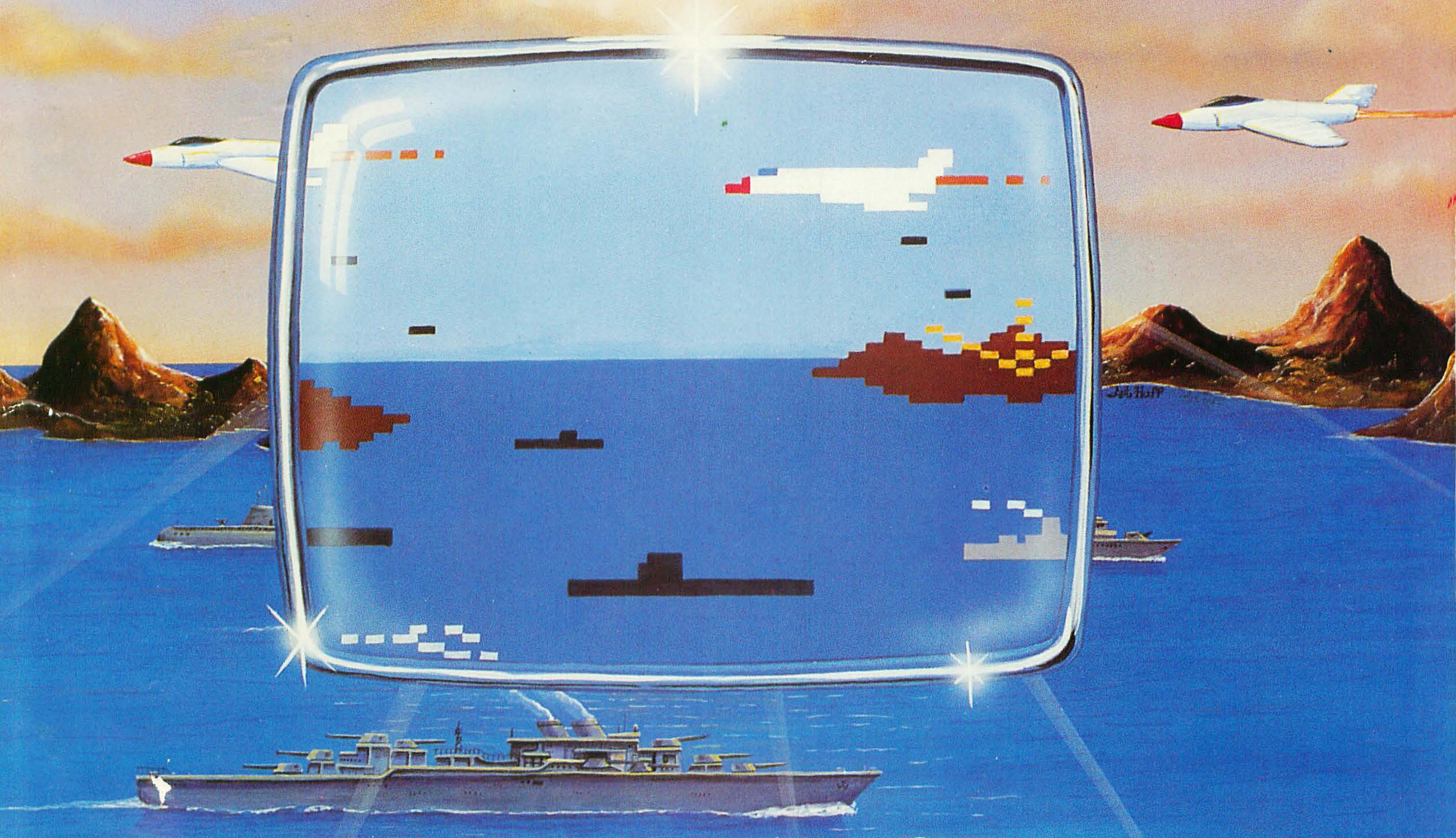
$14.95

GAMES ATARIS PLAY



By Hal Glicksman and Kent Simon

Based on Games Apples Play by
Mark James Capella and Michael D. Weinstock

Learn programming the fun enjoyable way . . . by gaming! Included is a vast selection of classic games for your Atari 400/800/1200 written in Atari BASIC. Why make programming hard work?

GAMES ATARIs PLAY

GAMES ATARIs PLAY

by

Hal Glicksman

and

Kent Simon

Based on Games Apples Play

by

Mark James Capella and Michael D. Weinstock

Cover Art and Illustrations by

Art Huff

DATAMOST™ INC

8943 Fullbright Avenue
Chatsworth, California 91311-2750
(818) 709-1202

DATAMOST

ISBN: 0-88190-118-0

ACKNOWLEDGMENTS

To my wife, Genie, for all those times she turned her head to see me sneaking up the stairs to the computer room.

Mark James Capella

To Diana and Aaron. Diana for her patience, understanding and the cold dinners she ate while waiting for me to finish just a little bit more. Aaron for the thrill of understanding what it means having a four year old around when things just aren't working correctly.

To Brian Davis for his creative artistic talents that greatly assisted our effort.

To Dave Gordon because he is Dave Gordon, a trusted friend.

Michael D. Weinstock

To Mark James Capella and Michael D. Weinstock for preparing an excellent collection of games and allowing them to be adapted to the Atari Home Computer.

To Kent Simon for opening his bag of programming tricks and putting some of his best POKEs and PEEKs in the programs.

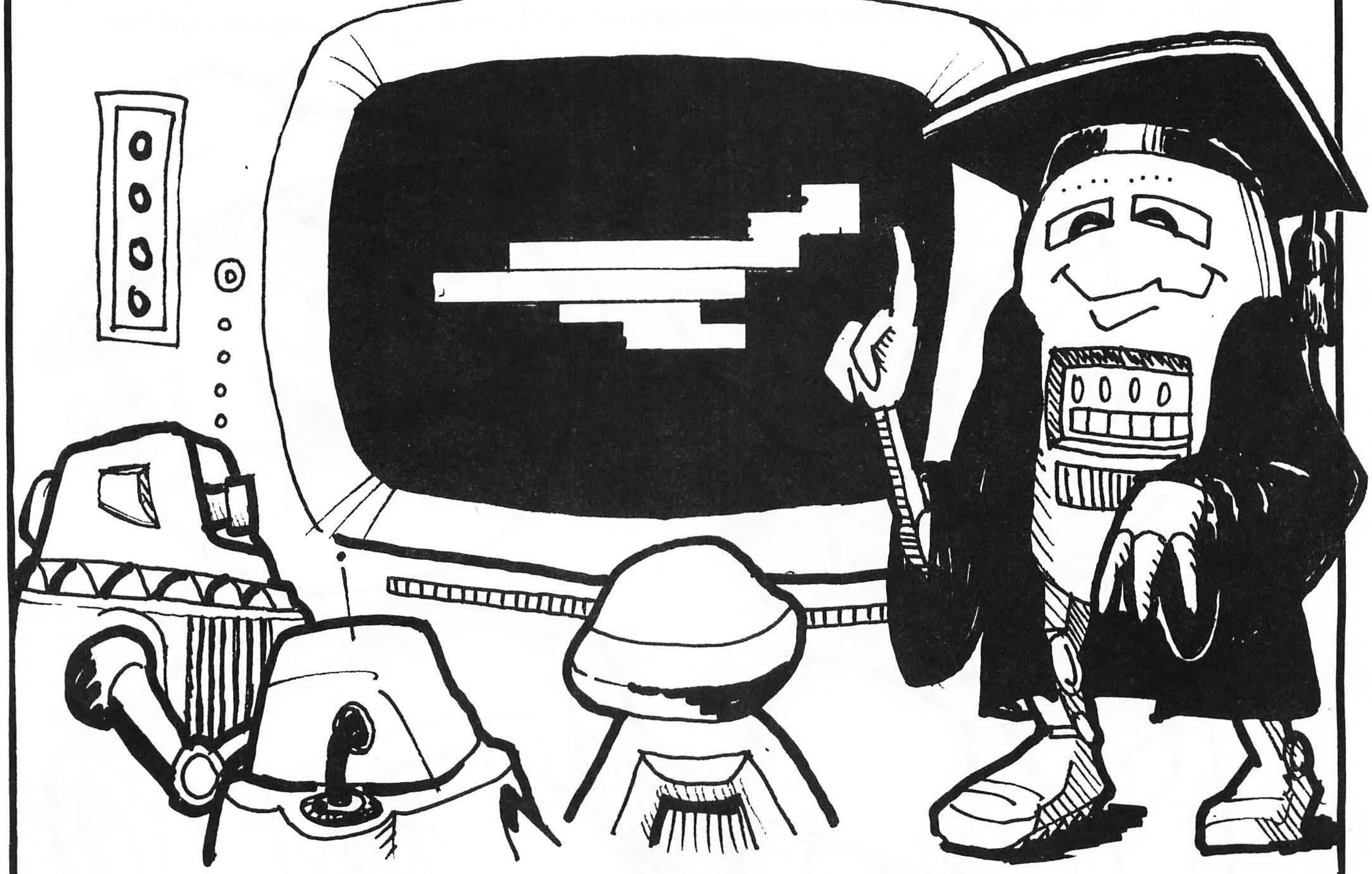
To Marcia Carrozzo, Director of Documentation, for allowing us to make lots of Atari sounds and play games while the rest of the staff tried to proofread manuscripts.

To all the DATAMOST staff who added their energy in getting this book completed.

Hal Glicksman

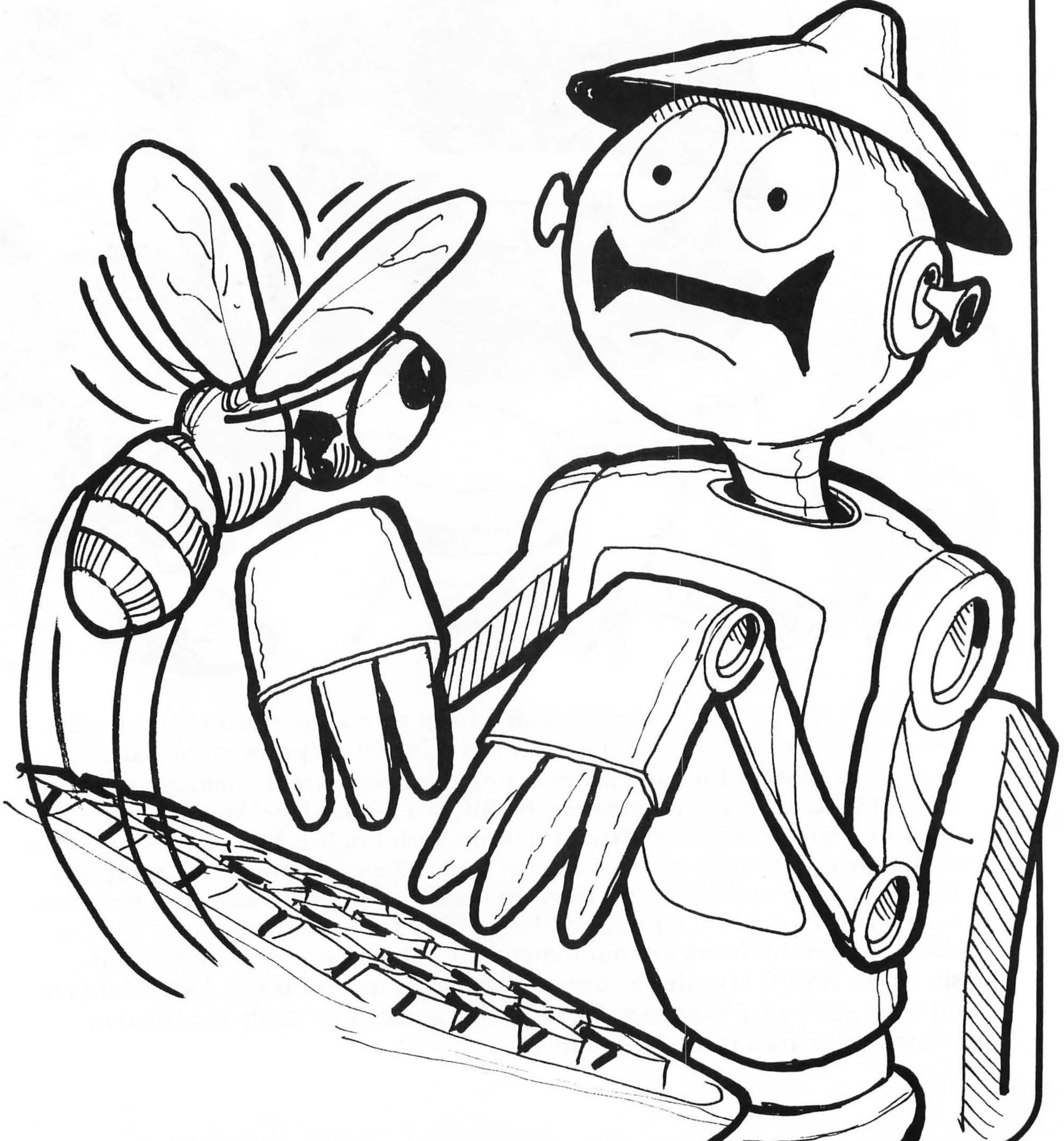
Table of Contents

INTRODUCTION
Using games to learn BASIC



The Atari Home Computer descended from several generations of arcade machines. The machine language high resolution graphics in the cartridge games are a lot flashier and faster acting than comparable images created from BASIC language commands. BASIC is a higher level language that allows programmers to write instructions with English-like statements and a relatively easy to follow structure and logic.The computer then interprets the commands and translates them to machine language when the program is run. BASIC is itself, a program designed to run other programs. Most of the games in this book are not dependent upon arcade-type action and speed, so BASIC is entirely appropriate and efficient for the task. Word and number games are easy to write in BASIC and can be easily modified to contain your own rules or examples.

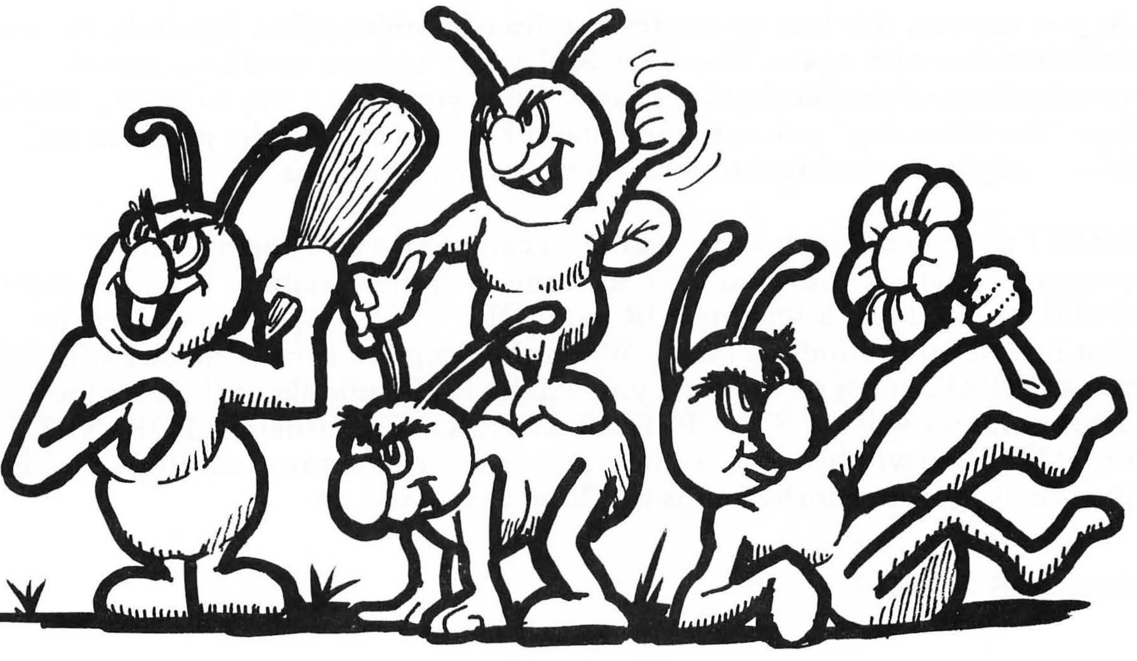
After learning how the games in this book work, you'll be able to write educational programs, guessing games, adventures, as well as beautiful color graphics and music programs. The mechanics of the games are explained in a way that will let you easily modify and customize the programs to your heart's content. Techniques are given that will allow you to examine the games inner workings and see what makes them tick. The games themselves are hours of fun. You will learn a lot by typing in the listings and seeing how typing errors show up in the programs as bugs and can then be easily chased out.



Do not be afraid of your computer! Remember, you are smart and the computer is dumb. The reason that programs have bugs is that the computer has to be told every little thing. It can't figure out a misspelled word or guess what you meant to type the way a person can. Fortunately Atari BASIC was written by some clever humans who programmed the computer to forgive as many errors as possible, and to point out and explain the errors it does find. Relax and have fun. NOTHING you type in from the keyboard can hurt the workings of your computer, you will just get what is called GIGO — garbage in — garbage out.

FINDING BUGS AND CHASING THEM AWAY

The Atari will find and identify two kinds of errors for you. Syntax errors are spotted immediately and pointed out by an ERROR- flag and an inverse (black on a white square) character over the error; however, the actual error might be quite a bit before the flag. A close parenthesis would get a flag if the open parenthesis were left out earlier. Run time errors are spotted when the program excecution is interrupted by an incompatability. The ERROR message will give a message number and the line the program stopped on, but the error will usually be in an earlier line. You have to figure out from the type of error and the line in which the error occured what needs to be done earlier in the program to correct the error. Finally, the program may be free of errors and still not run correctly. This is the final debugging and certainly the hardest part.



THE BASEMENT WORKSHOP

All the programs have unused numbers at the beginning of the listing that can be used to set up little routines for testing lines before they are used in the program. If the program is already entered, you can renumber a line by holding down the CTRL key, moving the cursor next to the line with the arrow keys, and typing in a new line number. This will *NOT* delete the original line, but will make an exact copy of the line at the new line number. The description for Sci-Fi tells how to set up a little music routine in the 'basement' that checks out and plays the music for the program.

Let's look at an example from Biorhythm. Line 2070 is the kind of line that is as clear as the bottom of Lake Erie.

```
2070 P3=INT(11.5+10*SIN(P2*6.283/23))
```

Renumber this line as 20. There are only two variables, P3 is the only one to the left of the = sign, so whatever happens to P2 has to effect P3. Try a FOR=NEXT loop to plug a series of numbers into P2 and print out the result.

```
10 FOR P2=1 TO 40
20 P3=INT(11.5+10*SIN(P2*6.283/23))
30 PRINT "P2=";P2,"P3=";P3
40 NEXT P2
```

As you can see, this line generates a series of numbers that diminish, increase and then diminish again. These are SIN waves and are used to create the curving lines of the biorhythm charts. Once you have a way of seeing how a line "does its thing" you can understand how it works in the program and how it might be modified to work in your own programs.

PRINT statements such as we just used can be put into the middle of programs to show you what is being done with the variables. They are most useful if placed just after the FOR in a FOR-NEXT loop, or just after the first line of other kinds of loops. When the program is running and you press BREAK, most of the time you will be in the middle of the play loop. The computer will say STOPPED AT LINE XXXX. Usually a FOR, NEXT or GOTO line will be close to the number where the program stopped. The line we used from Biorhythm is in a loop that starts at:

```
2030 FOR T=1 TO Z
```

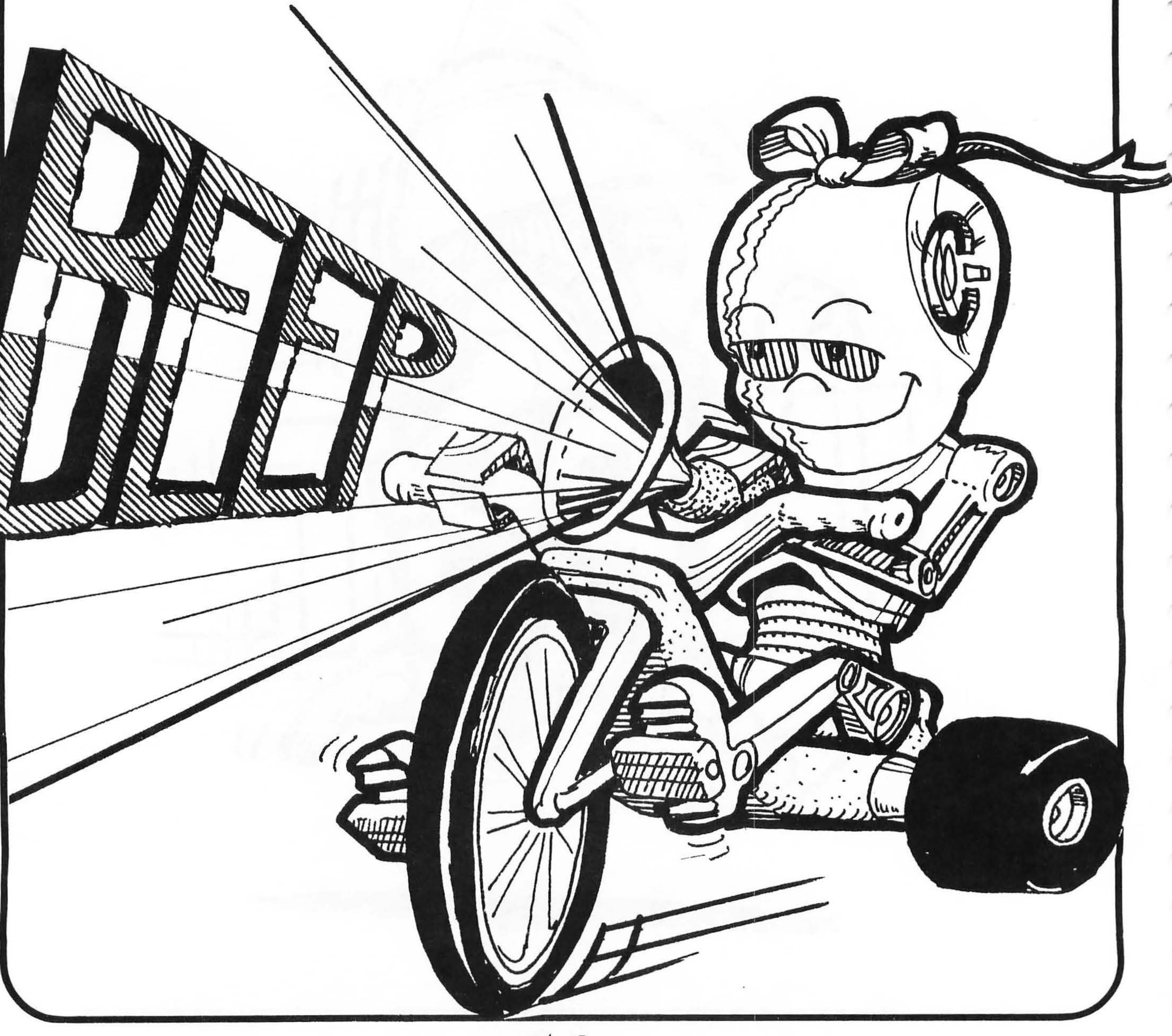
We can watch our variable P (physical state) working with the T (time) variables in the loop each time it is executed:

```
2032 PRINT "T=";T,"P1=";P1,"P2=";P2,"P3=";P3
```

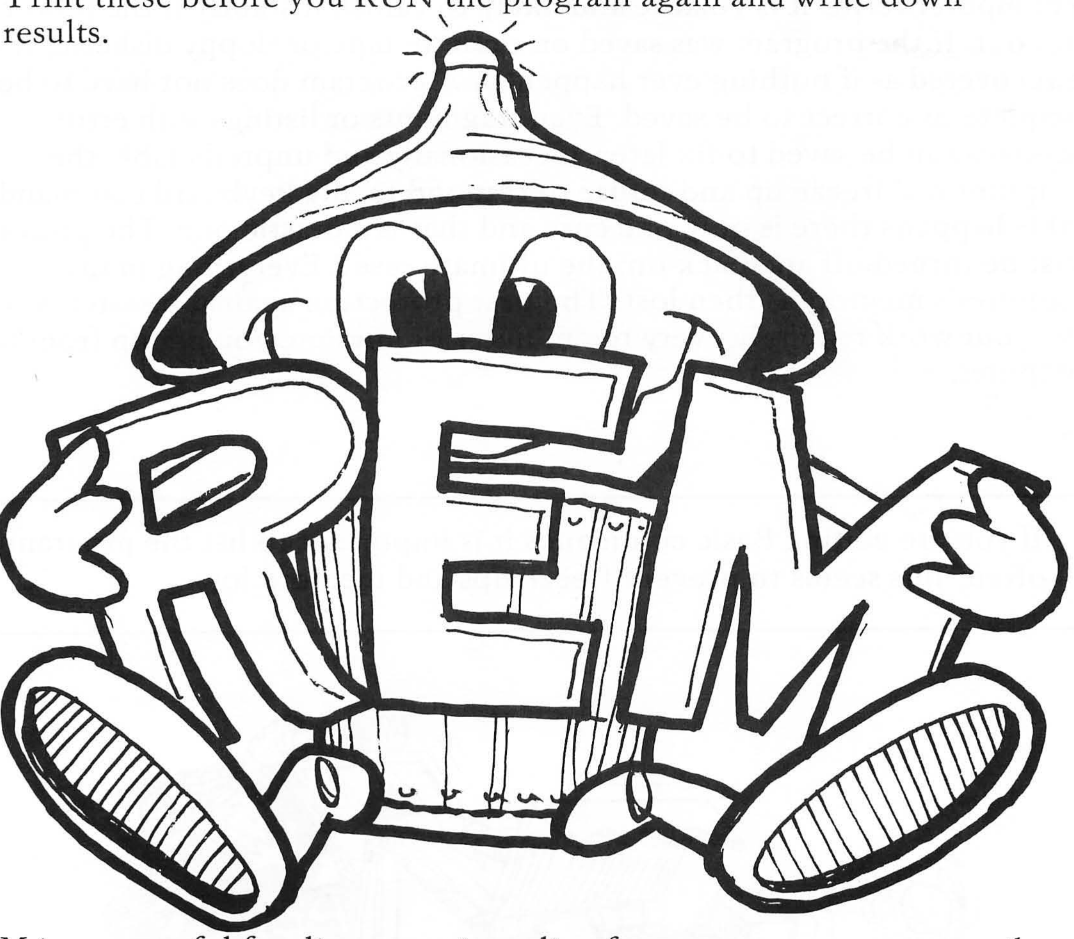
If the loop prints out the variables too fast for you to read, you can stop the works with an INPUT statement. DIMension a variable in the beginning of the program and put in an INPUT where you want the program to stop.

```
2034 INPUT AN$
```

Now the loop will print out the variables and wait for you to press <RETURN> before looping around again. Those pesky variables are the hardest things to understand, so many of them can be temporarily frozen by putting in a constant in place of the variable. This is particularly effective in graphics to show what variable creates what part of the image. If you see a line like COLOR X, you can change it to COLOR 2. Now some part of the graphic that used to change color will stay blue and you will know what action the variable applies to. Variables can also be arbitrarily entered into the program with input statements. In the example from Biorhythms, you could read the variables in line 2032 and then change them in 2034 INPUT T,P1. The next time the loop cycled you could see what your values for T and P1 did to the loop. If the program crashes once in a while, chances are that some strings



are getting too long or variables going out of range, (trying to draw a line too long for the screen is a common example). When the program has stopped you can still print out the variables and strings by typing ?P1 ?Z ?AN$ etc. Print these before you RUN the program again and write down the results.



REM is very useful for disconnecting a line from a program, temporarily. Move the cursor to the first position past the line number. Press CTRL and 3 INSERT's, then type in REM. What does the program do without this line? Type in an alternative line of your own devising using an intermediate line number. You can also isolate parts of programs with a GOTO. This is useful for temporarily eliminating a lengthy title screen while you are working on the play part of the program. STOP can also be used to test part of a program. Put in a STOP command wherever you want to tinker with variable. After the program stops you can print out the program's variables and enter your own variable and strings in direct mode

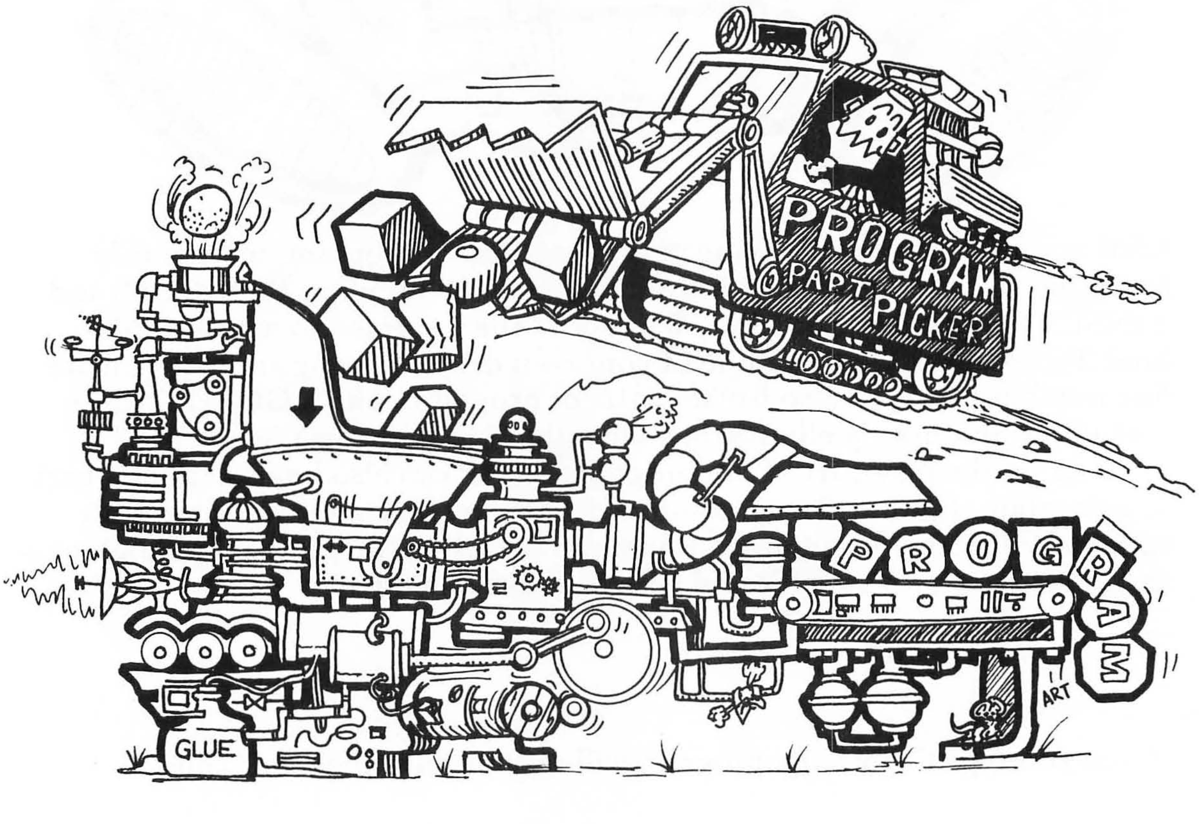
```
A$="horse"
X=3
```

When you type CONT the program will resume with these variables.

SAVING YOURSELF FROM DISASTER

Whatever is typed into the computer's memory exists in a temporary state. In computer terms it is volatile and likely to vanish instantly if the power goes out. If the program was saved on cassette tape or floppy diskette, it can be recovered as if nothing ever happened. A program does not have to be complete or correct to be saved. Even fragments or listings with error messages can be saved to fix later. Occasionally and unpredictably the computer will freeze up and refuse to respond to any keyboard commands. If this happens there is only one cure and that is a drastic one. The power must be turned off and back on; the ultimate reset! Everything in the computer's memory is then lost. The best protection against disaster is to save your work regularly every ten minutes or anytime you get up from the computer.

> If you are editing Basic commands it is important to list the program often; this seems to prevent freeze-ups and memory loss.



We emphasize saving programs often because we want you to play around with the programs, try whatever comes into your mind and mess around with every line. Put in silly routines and eventually get the program so messed up that you have to toss it out and load in a previous version. This is a good way to learn, and certainly the most fun of any method of learning Basic.

If you have a program running half-way right it is a good idea to save it as an intermediate version. Often we make some useful modifications and then mess the program up with later additions. The useful parts might get thrown out with the mess. The Atari disk system allows programs to have suffixes added to the names such as FUNGAME.V1, FUNGAME.V2, etc. This makes it easy to identify the version. Get used to starting each programming session by putting today's date at the beginning of the program, you will thank me for it later. When the program is finished the disks or tapes with obsolete versions can be reused for other programs.

The Atari can save program listings as text files for easy editing and merging with other programs. If you have written a nice title screen and would like to modify it for use with another program you can file away just the lines of the title screen portion of the program:

```
LIST"D:GAMENAME.LST",1000,1540
```
(D: for disk drive, C: for cassette recorder)

Retrieve this section of program with:
```
ENTER"D:GAMENAME.LST" (C:for cassette)
```

Once a section is isolated it can easily be merged with other parts of programs by additional ENTER statements. If any line numbers are the same, the incoming line numbers will replace the existing line numbers. Short segments can be quickly renumbered to avoid conflicts. ENTER does not clear out existing memory, RUN and LOAD replace everything in memory.

USING GAMES TO LEARN BASIC

We recommend that you develop your gamesmanship in three stages:

1) Copy the listings in this book and get the games to run.

2) Learn the function of BASIC commands by modifying the games and writing variations of your own.

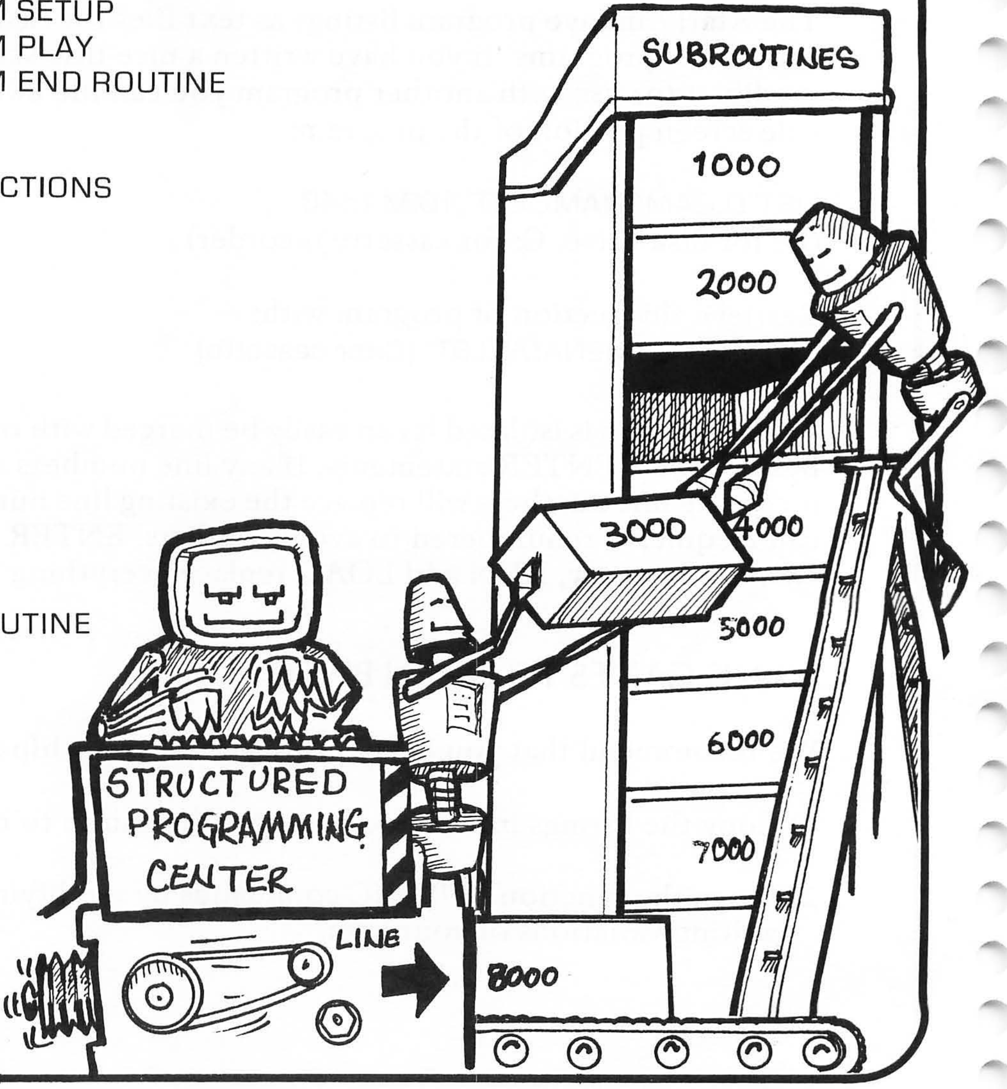
3) Use the structure program and routines from our games to construct new games of your own design.

4) Don't stay up too late or neglect your family. When it's not fun any more, take a break.

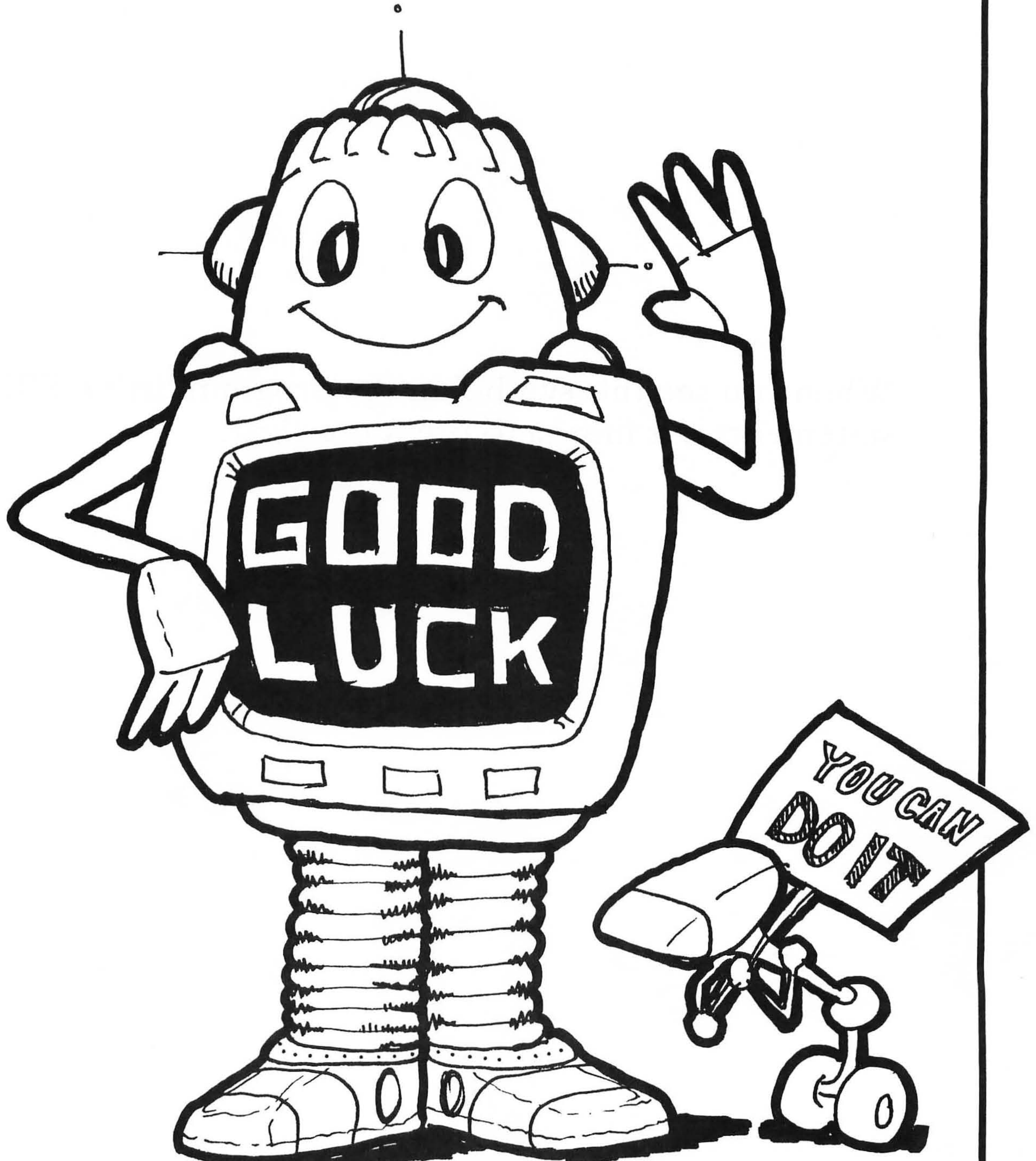
STRUCTURED PROGRAMMING:

Some of the programs have been written in Structured BASIC with a standard framework that can be used to develop your own programs.

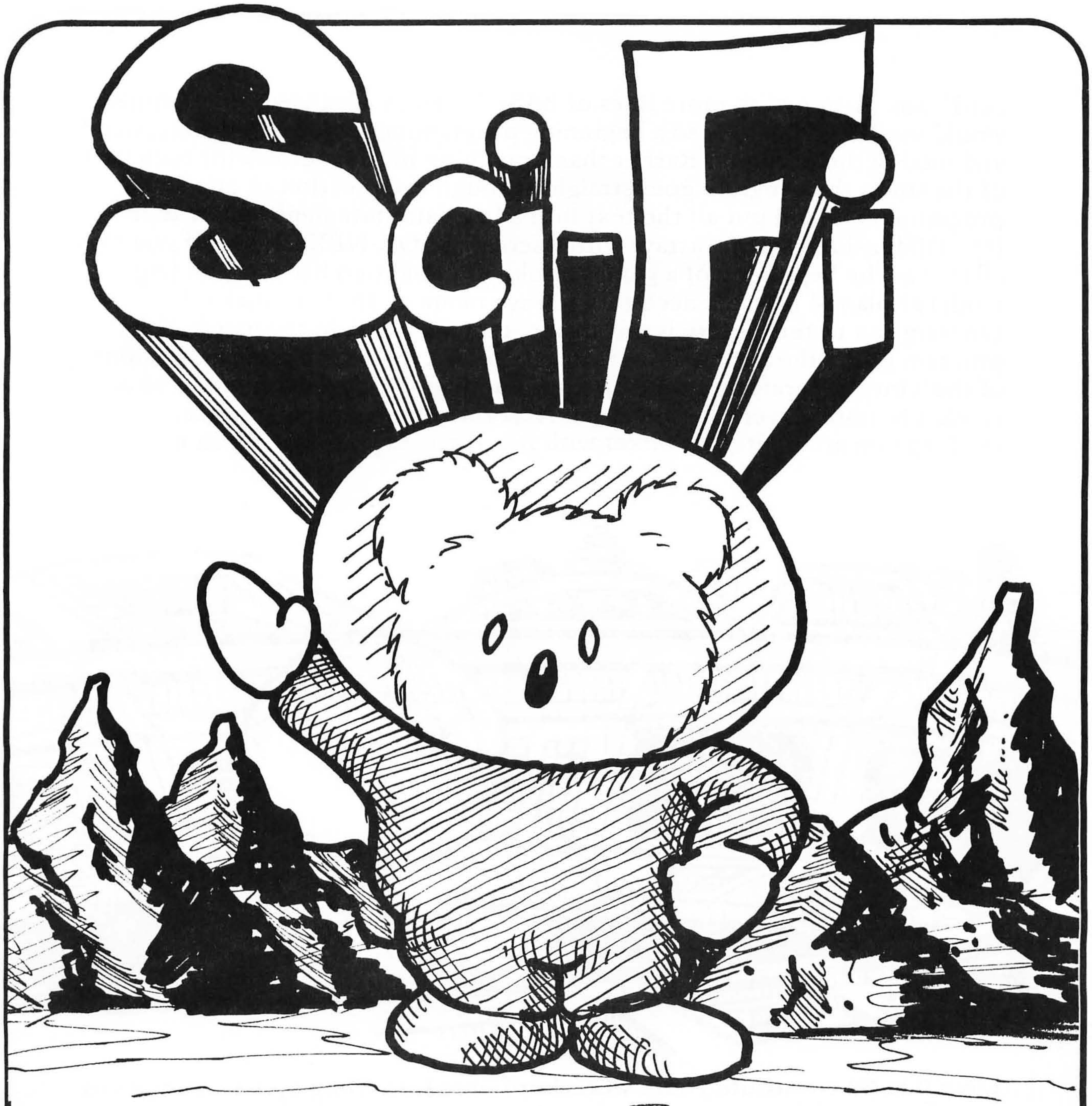
```
100 REM **************************************
110 REM *     TITLE OF LISTING IN A BOX        *
120 REM **************************************
200 REM DIMENSION VARIABLES AND STRINGS IN 200-290
300 GOSUB 1000:REM INSTRUCTIONS
310 GOSUB 2000:REM SETUP
320 GOSUB 3000:REM PLAY
330 GOSUB 4000:REM END ROUTINE
340 END
1000 REM ***
1010 REM *** INSTRUCTIONS
1020 REM ***
1990 RETURN
2000 REM ***
2010 REM *** SETUP
2020 REM ***
2990 RETURN
3000 REM ***
3010 REM *** PLAY
3020 REM ***
3990 RETURN
4000 REM ***
4010 REM *** END ROUTINE
4020 REM ***
4990 RETURN
```



This program will run even though it doesn't do anything. You can use it for your own programs to help stay organized and remind you of what needs to be done. It is also useful psychologically. When you are inspired you can work on the play section. When you are tired you can work on the title screen. When you are energetic you can write the setup. When you are angry you can put fiendish maledictions in the end routine.

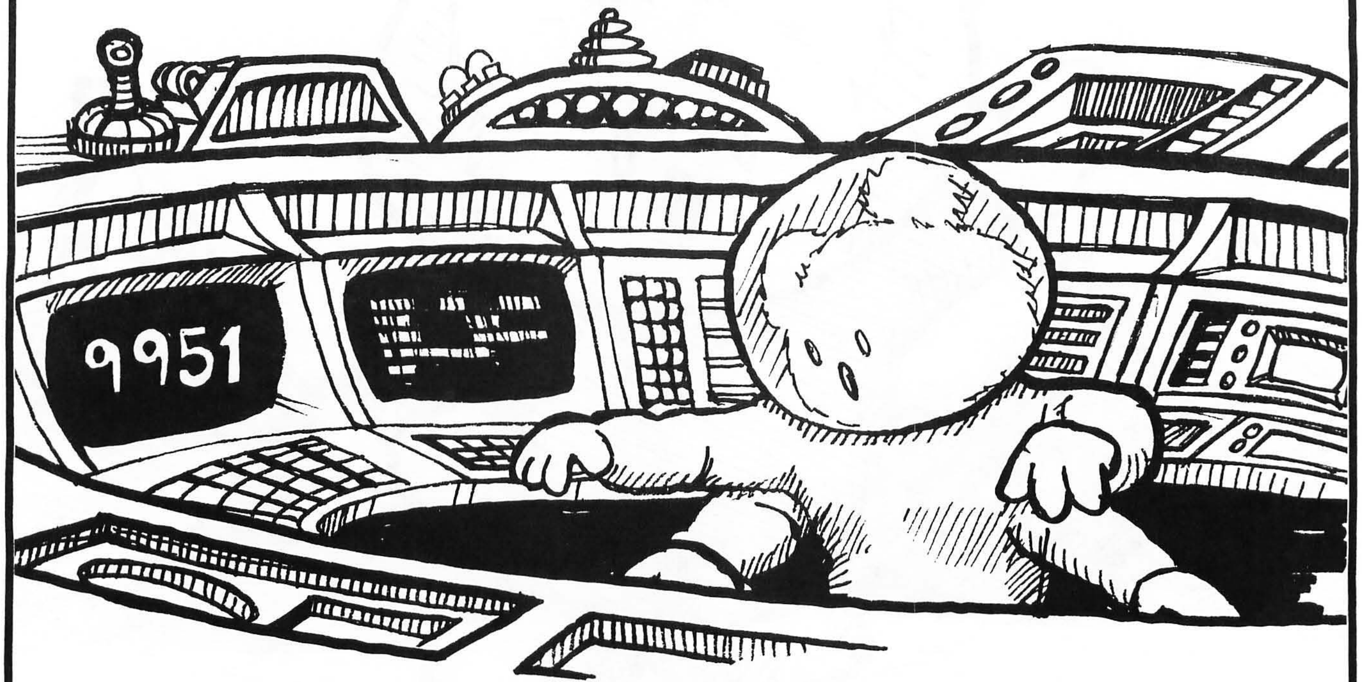


When you see this symbol in the program listing PRINT statements }, it means to press esc clear.



This program is an entertaining collection of short stories which you can make as personal as you wish. Every time you run the program you are given the chance to enter your own list of names and places to use in the stories. Once you see how the program works you may want to write your own version of the actions and sequence of events. This program was chosen for a line-by-line description because most of the routines for handling text and data are introduced here. After you have written out this game you will be able to write just about any type of quiz, educational program, or word game.

Sci-Fi was written with more lines of BASIC than an advanced programmer would use. This was done so a beginning programmer could easily understand and modify the program. Rather than branching to subroutines for each part of the story, the program goes straight through each section. A good programmer would put all the text into a few data statements and read it into DIMensioned string arrays with a series of FOR-NEXT loops. If you saw all that at the beginning of a games book, you might go back to building model airplanes! It is not necessary to save memory space or make the program run faster if there is no 'arcade' type of action in the game. This program is like the shaggy-dog stories it creates. There are as many versions of the 'story generator' as there are programmers, and each one thinks his version is more clever than the next. As soon as you get this program working you are invited to tinker with it and create your own version.



Lines 0 to 99 constitute your "workshop" area for setting up test routines to try out parts of the program. This is where we have placed a routine that tests the data statements that play music. If you have typed the music data correctly, you should hear the "William Tell Overture." You can jump ahead to the music section in 3900 and try out some music before writing any more of the program.

Lines 100-170 are the introduction to the listing. REM statements are not executed in the program, rather they are used to leave notes to yourself and other programmers. If you save a partially completed game or one you have modified, leave yourself a note:

```
99 REM Part finished Friday 1/13
```

Line 220 sets all the variables to zero, a little bit of homework that has to be done to Atari BASIC programs.

Line 250 TRAP sets an error trap for the next line where the printer is called. If no printer is attached, or the printer is not ready to print, the error trap sends the program to line 300 where it will continue as if line 270 didn't exist.

Line 270 "OPENS" a device input-output path. The #1 path is assigned to output (8) with no (0) auxillary code and told that the output device is a "P" printer. Once this line is executed in a program all text can be sent to the printer with the command PRINT #1 "your words."

Lines 300-330 are the framework of the structured program that is used throughout this book. You should type in the entire structured program (see page 18) so the program will run as you add to it.

Line 340 follows the end routines in lines 4000-4990. Setting the keyboard back to upper case is a convenience to the user because the computer will be in BASIC when the game ends and all BASIC commands must be in upper case. This is the last line to execute before:

Line 350 END. The END statement doesn't have to be at the end of the program listing. In fact, END can occur in several different places in a program. You can write a command to end a program if a certain condition is met such as:

```
IF X = 20 THEN END.
```

Line 1000 is the target of the GOSUB in 300.

Line 1010 sets an invisible cursor so that a white square will not be displayed in front of all the text on the screen. We suggest you type:

```
1010 REM POKE 752,1:REM INVISIBLE CURSOR
```

REM can be used to temporarily disconnect any line from the program, even one with another REM statement. It is useful to keep the cursor visible while you write and test out the program. Delete the first REM when everything is working.

Line 1020 clears the screen. In order to put a function key command in the program, put the command in a PRINT statement and press ESC(ape) before the function command. CLEAR will print on the screen as a bent arrow and on the printer as a curly bracket.

Lines 1030-1090 formulates the title screen displayed in a box.

Line 1030 is the top of an outlined box. It does not appear on the printer output. The edit features of the Atari will really help you wiz through screen formats such as this. Start with the longest line, 1060. Use the (bar symbol) on the equal key for the sides of the box. Guess at the number of spaces to put into the center of the box. To run only this section, type GOSUB 1020. You can add or delete spaces to center the text "/Science-Fiction Tales/." Now comes the tricky part.

```
Type LIST
```

Press CTRL (UPARROW) until the cursor is opposite line 1060.

Press CTRL (RIGHTARROW) until the cursor is over the 6 on 1060 Type 3 and RETURN. This will duplicate 1060 as 1030. Repeat this process making all the lines from 1030 to 1100 into "/Science- Fiction Tales/." This will provide the PRINT statements, quotes, spaces, and line numbers for the entire box. It will then be quite easy to type over "/Science-Fiction Tales/" with the correct text, frame, or blank spaces. Press RETURN after each edit and DO NOT edit two different lines at once.

Move the cursor to the (bar symbol) in line 1030. Type (CTRL) Q for upper left corner and (CTRL) R for the horizontal line. Hold down the (CTRL) R and it will repeat. When you get to the ending (bar symbol) type (CTRL) E for the upper right corner. For line 1040 just space over the text to erase it. Change the number of line 1040 to create lines 1065 and 1075 Edit line 1070 erasing with the space bar until you get to the 'F' in 'Fiction.' Type 'from' over the existing text and blanks to the bar.

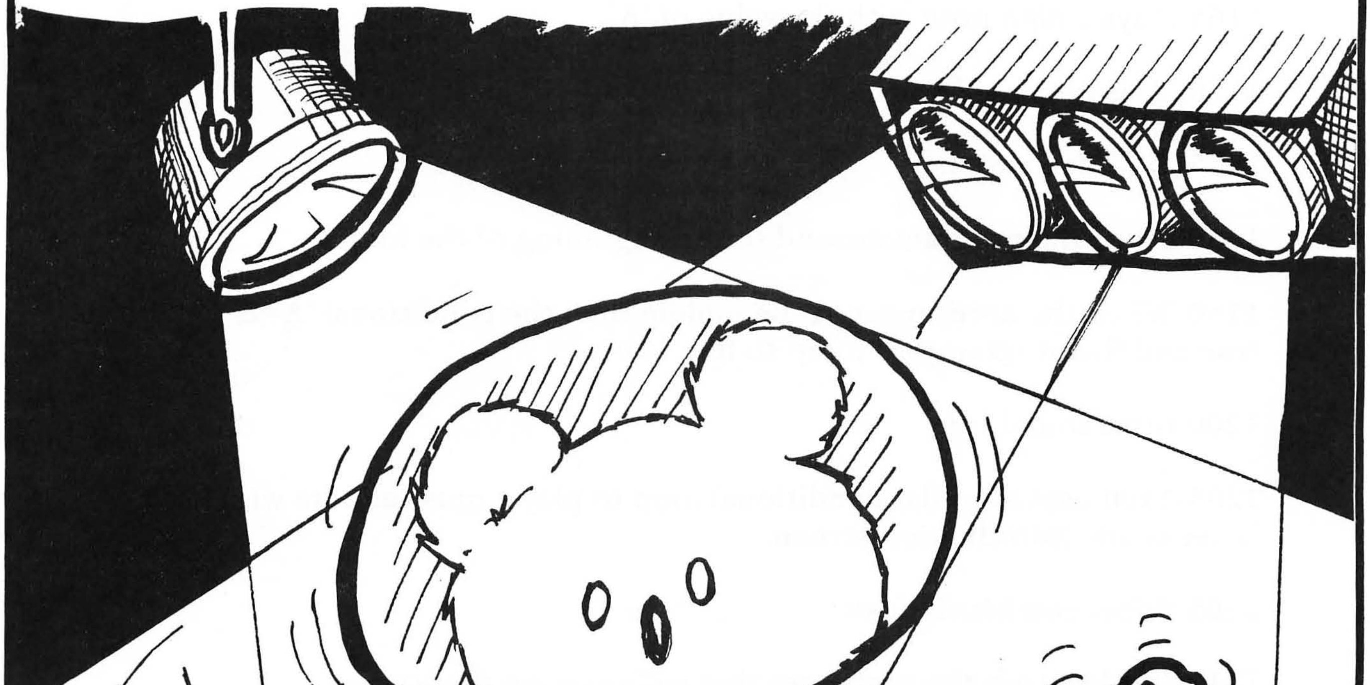
LIST and RUN the program as often as you would like to check your work.

This technique might be slow at first, but it will save you time if you completely learn the edit commands.

********SAVE YOUR PROGRAM*********

By the time you have reached this point, you have made a fair investment of time in entering commands. Save what you have done so far on tape or disk.

Always save what you have done before taking even a short break from the computer. It is also a good idea to list the program often; it seems to prevent loss of data and freeze ups that are known to happen after an intense period of editing.



Lines 1140-1200 make a little musical routine with a conditional loop. The 'B' variable is set to the lowest allowable number of the musical scale. 'A' is still at 0.

1155 uses 'B' to make a low note.

1160 is a one line FOR-NEXT loop that waits a fraction of a second.

1165 Plays a high note with the value of 'A'.

1170 'A' is increased and

1175 'B' is decreased.

1180 Sends the program around to the beginning of the loop.

1150 When the notes meet in the middle then the conditional 'A=B' will be true and the program will jump to line 1200.

1200 turns sound off

1205-1400 uses a similar conditional loop to play a musical note with each letter of the introduction screen.

1205 Prints two blank lines.

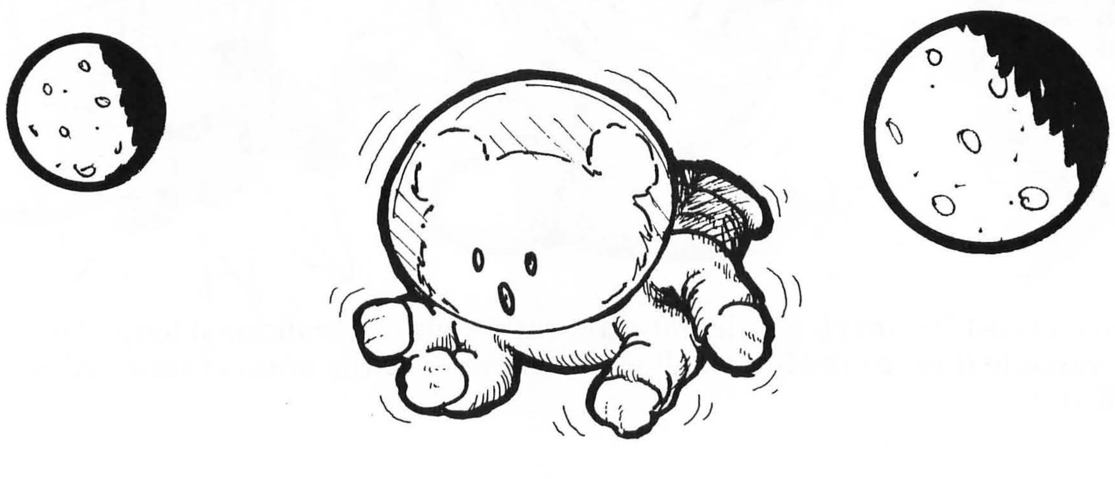
1210 and 1230 are the sentences that will print on the screen.

1300 Sets 'A' to a high note

1310 Is the conditional that ends this loop. The loop will end when a note has been played for each letter of '(E$)'.

1320 Plays a note for 'A'.

1330 Adds 1 to 'A'.



1340 Prints one letter of (E$) each time the loop goes by with another increment of 'A.' The semicolon at the end lets the letters follow each other.

1350 Waits a little.

1360 Loops back to 1310 where 'A' is tested to see if the loop is to continue. If you want to understand this loop better add a line:

```
1315 PRINT"A=";A
```

This will print out the value of 'A' as it is used in the loop. Take this line 1315 out when you are done experimenting.

1400-1500 is yet another variation on the conditional loop.

1410 B is set to one and is used to count the letters of (F$). 'A' is brought over from the loop above, and is use to bring the notes up the scale starting one note higher than the stopping place of the loop above that printed (E$).

1420 Is the conditional test to check if all of (F$) has been printed. If it has, then the sound is turned off and the program goes to line 1500.

1430 Plays the sound for 'A'

1440 Prints the 'B' letter of (F$).

1450 Takes one away from 'A' for the next higher note, and adds one to 'B' for the next letter of (F$).

1460 Waits a little.

1470 Completes the loop. Play around with these loops and sound routines until you have come up with some variations of your own. The main loop that prints out the Science-Fiction is another variation on the conditional loop, so it will help to understand these loops first.

1510 Asks for the player's name.

1520 Sets the keyboard to lower case.

Also serves to hold the title screen on until the player is ready to proceed.

1530 looks for the lazy player who just presses RETURN instead of responding to INPUT requests. This line names these players "Nameless Blob." It is important to put something in NA$ to prevent a crash later on.

1540 to 1640 asks which list of names the player wants to use. The answer is evaluated by the IF statements and assigned as a value of HG. If you choose the program's list of names then line 1630 sends the program around the entire input section directly to 'PLAY.' There is no error to check for an empty input statement. If you press RETURN without answering the question the program defaults to 'C' (both lists of names).

1990 Ends the title section and returns to 120 where it is sent back to 2000

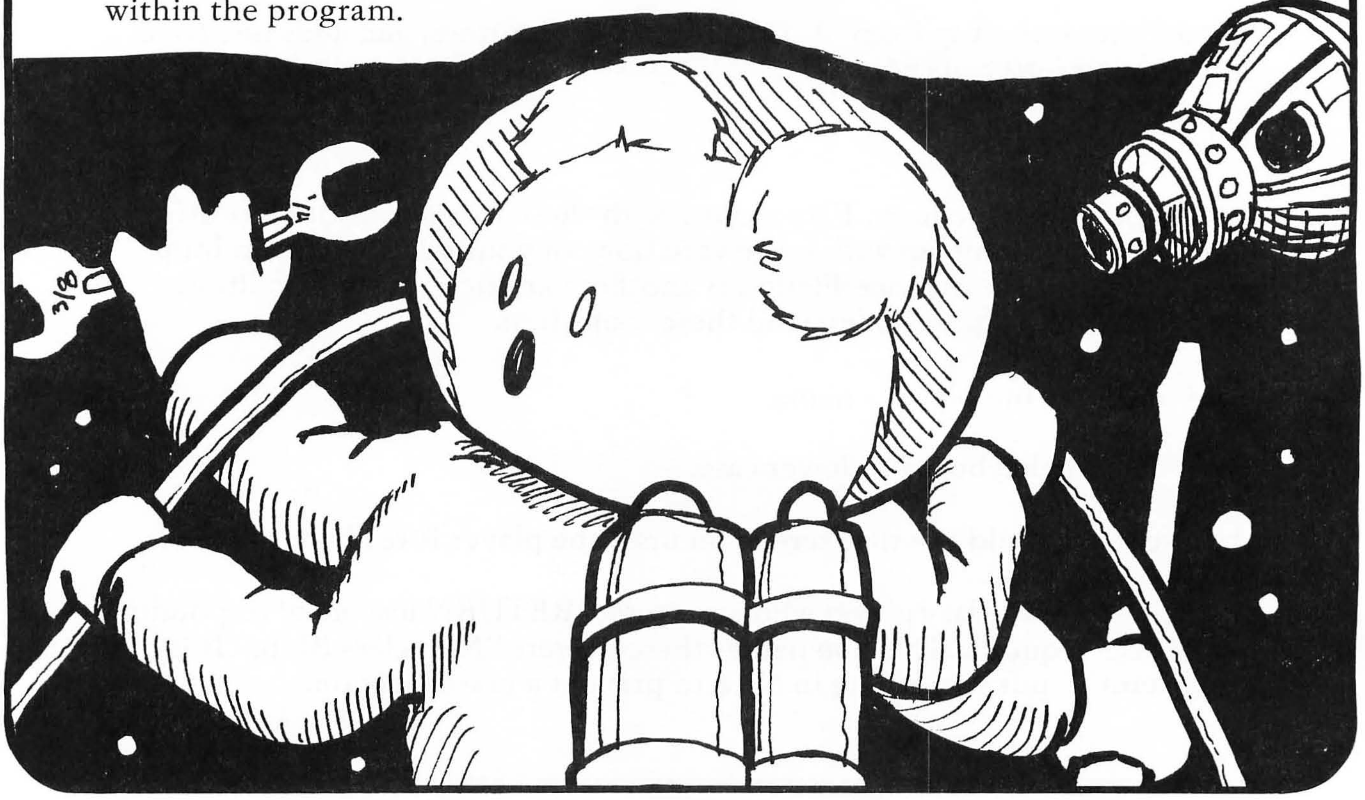
2000-2557 This entire section is a table of string variables with an input routine for the player to enter some of the names in each section.

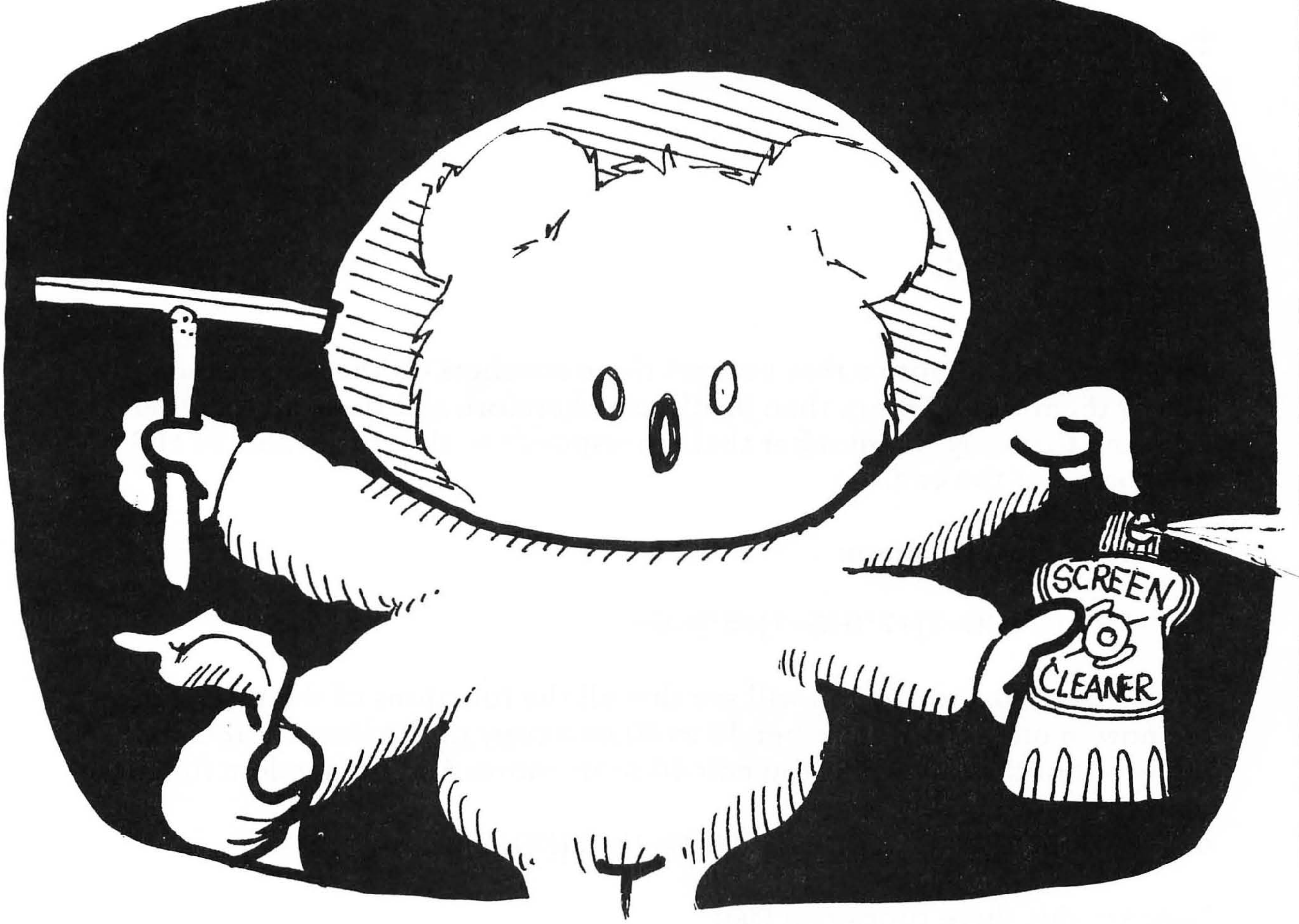
2020 Clears screen

2025-2045 Dimension those variables!

2055 Prints the top of a box to use as a guide for the length of the player's input. The box does not show in the printed listings but will appear on the screen listings. Just renumber line 1030 and take out the spaces before the first corner symbol.

Line 2200 represents the first name provided by the program. The spacing of the line numbers is important. Each group starts at 00 or 50 and is numbered by five. There are seven names in the program and five places for player input. These proportions can be different, but must be consistant within the program.





Line 3000-3020 sets the numeric variables to 0 before you use them, and then clears the screen.

Line 3200 is a random number which gives a value of 1-5 to 'R'.

Line 3205 the ON-GOTO selects from the titles in lines 3210-3230.

3255 Take a deep breath, place your right forefinger on your forehead and repeat three times "THIS IS NOT MATH." Now that your brain is energized we will explain that this is the routine that randomly selects either the program's choices (0-6), the player's choices (7-11), or both (0-11) according to the selection of HG made in line 1610. The tricky part is that 12 * (HG=0). HG=0 is a truth test that has the value of 1 if HG=0 and 0 if HG= any other number. In order to construct a big gee-wiz conglomeration like this we build up the parts in short segments, make them work, and then put the parts together. Lines 0 to 99 can be used for experimenting on the program, and when the line works it can be numbered to go where it belongs.

Try this:

```
10 HG=Ø
20 INPUT HG
30 PRINT 12*(HG=Ø)
40 PRINT 7*(HG=1)
50 PRINT 5*(HG=2)
60 GOTO 20
```

Run this part and notice that you get three numbers on the screen,and only one of them will be more than 0. We can therefore add these all together knowing that only the number that corresponds to the truth value of HG will come out the end.

Change the test program:

```
40 PRINT 12*(HG=Ø)+7*(HG=1)+5*(HG=2)
```

Delete lines 30 and 50, you will see that all the functions of the three lines are now in one line. Renumber 40 as 80 so a copy of the line will remain safely out of the way while you edit 40 some more. Add the random function:

```
40 R=INT (RND(Ø)*(12*(HG=Ø)+7*(HG=1)+5*(HG=2)))
```

Now say this three times real fast:

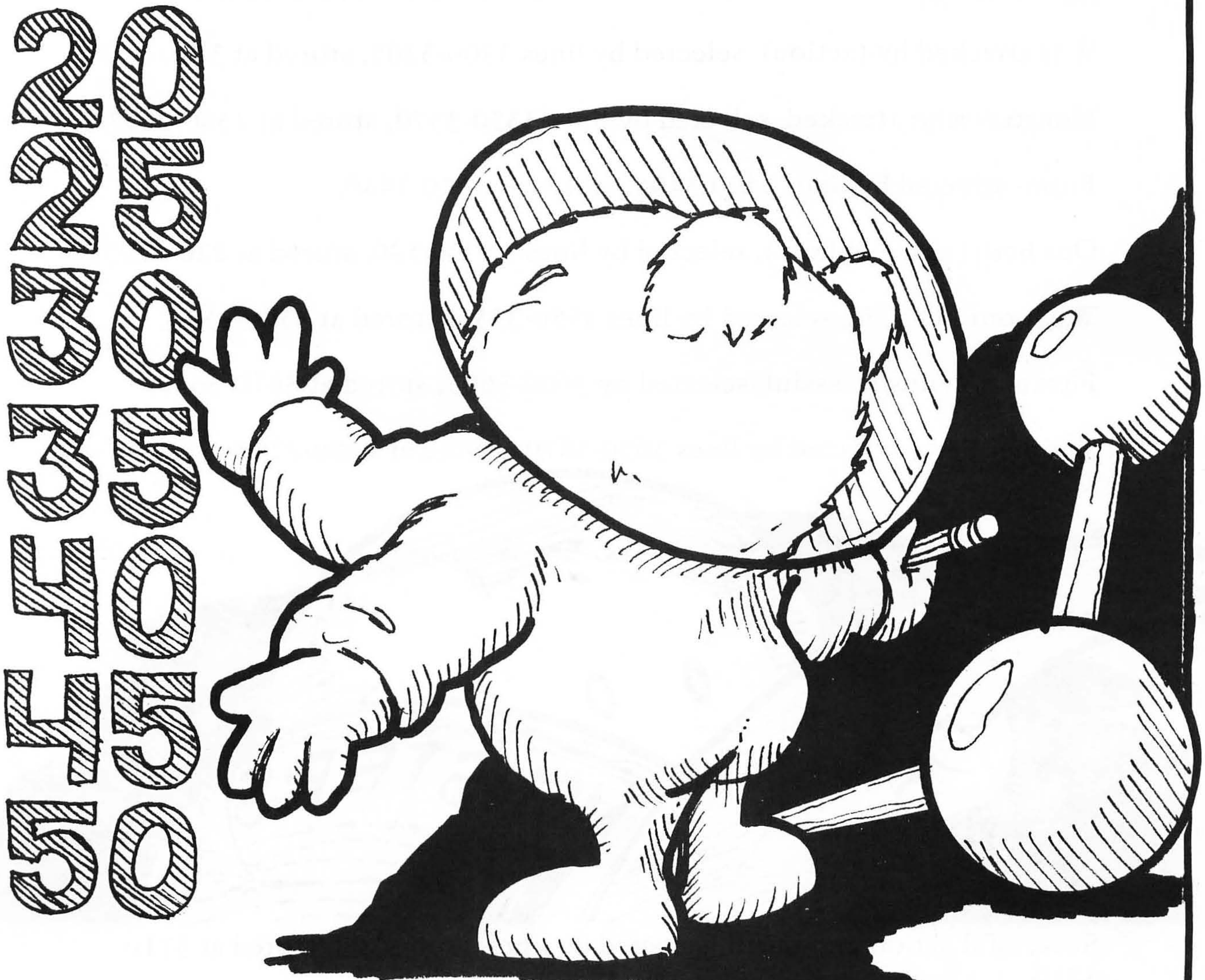
Proliferating parentheses plague perfect programs.

Add a routine to run the random function 20 times.

```
30 FOR I=1 TO 2Ø
50 PRINT "R=";R
6Ø NEXT I
7Ø GOTO 2Ø
```

Come back! You are not through yet. The HG=2 variable is supposed to yield a number from 6 to 11, so 7 must be added to the value of R after the random function is run. This is done by adding +(7 * (HG=2)) to everything after R. It takes two more sets of parenthesis and much fiddling. Just remember to renumber the last working version so you can start over if you mess up. Always add parenthesis in pairs, although the left one doesn't always have to come first. Now an input of 0 should yield random 0-11, 1 should yield 0-6, and 2 should yield 6 to 11.

Now put parenthesis around that whole statement and multiply it by 5* to get lines numbered by 5.



Line 3260 sends the program to the group of names starting at 2400 plus the random number generated above. The name of the "Place that will be attacked" is stored in A$.

Line 3270 sends the program to 3800 where A$ is printed out with musical accompaniment.

The program repeats the random selection of parts of the story in the following sequence:

Places that will be attacked, selected by lines 3255-3270, stored at 2400-2457.

Was attacked by-(action), selected by lines 3300-3305, stored at 3310-3330.

Monsters who attacked, selected by lines 3350-3370, stored at 2300-2357.

From- selected by lines 3400,3405, stored at 3410-3440.

Our hero (name of hero), selected by lines 3500-3520, stored at 2200-2257.

What our hero did, selected by lines 3550-3555, stored at 3560-3580.

First action unsuccessful, selected by 3600-3605, stored at 3610-3630.

Another hero. selected by lines 3650-3670, stored at 2200-2257.

Successful action and result, selected by lines 3700-3705, stored at 3710-3730.

Since each section of the story is printed out by GOSUB 3800, the program jumps from the last line to the end routine starting at 4080.

Line 3800 is the print-out with musical accompaniment.

Line 3805 checks to see if the random routines have chosen empty strings. If so, the player's name is substituted for A$.

Line 3810 sets the variables for the letters and notes to 0.

Line 3815 prints the next letter of A$. Notice the semicolon after A$.

Line 3820 reads a note and a time value from the data tables of notes staring at 3910.

Line 3825 plays a sound with the note just retrieved.

Line 3830 looks at the keyboard cursor location with PEEK (85). If the cursor is past 30 and the next letter is a blank then the computer goes to the next line. This is a simplified 'wrap around' technique.

Line 3835 restores the data tables to the beginning. Be sure to put in a 0,0 for the last note, this command will replay the music.

Line 3840 adds one letter to the A$ counter so the next letter will print when the loop gets back to 3815.

Line 3845 is a delay loop set by the time value of the note.

Line 3850 turns off the sound so that notes of the same value will not play as a continuous sound.

Here's a little music box that you can build in the basement workshop (lines 0-99) to test out the music routine and the data statements without having to run the whole program each time you want to hear music. Clear out any old lines and enter the following program:

```
10 NT=0: TM=03
20 READ NT,TM
30 IF NT=0 THEN RESTORE
40 PRINT "NOTE=";NT,"TIME=";TM
50 SOUND 0,NT,10,10
60 FOR I= TO (TM*20):NEXTI
70 SOUND 0,0,0,0
80 GOTO 20
```

The loop is much shorter in this segment so TM has to be multiplied by a larger number. Enter the data statements in lines 3900-3990. Even though the music box program never goes past line 80 it can read the data statements anywhere in the program. Line 40 prints out the notes and time. If there is a sour note or time error hit, the break key and check the values on the screen. If you left out a comma, the time and notes will be switched with very strange results, but this error is easily traced and fixed.

The music box can be left in the program and turned off with a line:

```
5 GOTO 100
```

Line 3855 this is the test for the conditional loop. The program will loop to 3815 until PT equals the lenth of A$ and every letter has been printed. When PT is no longer true the program drops down to the next line

Line 3860 returns from the GOSUB to the next section of story selecting.

Line 3900 stores musical notes as data statements. The musical routine is a very simple one with only one 'voice.' The program can easily accommodate additional variables for the sound command if you want more sophisticated music with your stories.

Line 4000-4030 clears the screen, says goodbye and returns to 140 where the program ends.

Line 4080-4310 is a routine that is called in 3735 to offer options after the story is on the screen.

Line 4120 accepts a response to the choices and stores it in G$.

Line 4130 looks for an empty string and assigns it a value of "NO." Since this is not one of the choices looked for in the next two lines the program will 'fall through' to 4190 and ultimately 3000 where another story will start.

In line 4140, the GOTO statement will send you to statement 4200 if you request a printed copy. We just checked the first letter of A$ to see if it was P or p. We had to test for and eliminate the empty string in line 4130 or there would be an error if A$(1,1) did not exist.

Line 4180 checks to see if you want to exit the program. Here we also check for either a capital or lower case 'x' in the response.

Line 4190 returns to 3000 to start a new story.

Lines 4200 and 4210 look at the screen memory locations where the text is displayed.

Lines 4220-4270 set up a nested loop. There are 40 letters (0-39) numbered 'Y' on each of 12 lines numbered 'X.' This prints only the top 12 lines on the screen.

Line 4240 looks for any special Atari characters outside the regular ASCII alphabet and adds 32 to any that are found. This will convert the characters so they will print on an ASCII printer.

Line 4245 prints out each character to device #1 opened in line 260. We could have written the program to print A$ as it was sent to the screen, but this method allows you to read the story first and throw it away without wasting paper. There is no error trap to warn users that a printer is not hooked up. If you have friends like that, you can easily add in a trap routine.

Lines 4300-4310 clear the screen and sends the program back for another story after printing.

Line 4990 is a fossil left over from the structure program. It never gets used.

```
5 REM MUSIC TESTING ROUTINE HERE
10 DIM A$(80),B$(20),C$(20),D$(20)
15 DIM NA$(30),E$(200),F$(200),G$(20)
20 GOTO 100:REM DELETE THIS LINE TO TEST MUSIC
25 NT=0:TM=0
30 L=15:REM SETS LENGTH OF NOTE
35 READ NT,TM
40 IF NT=0 THEN RESTORE
45 SOUND 0,NT,10,10
50 FOR I=1 TO (TM*L):NEXT I
60 PRINT "NT=";NT;"  TM=";TM
70 SOUND 0,0,0,0
80 GOTO 30
85 REM LIST 3900,4000 TO SEE NOTES
90 REM TYPE 20 GOTO 100 TO RUN SCIFI
100 SETCOLOR 2,5,14
110 REM *************************************
115 REM *                                   *
120 REM *          SCI-FI ATARI             *
130 REM *       BY HAL GLICKSMAN            *
140 REM *          APRIL29,1983             *
150 REM *      COPYRIGHT DATAMOST           *
155 REM *                                   *
160 REM *************************************
170 REM
220 A=0:B=0:HG=0:R=0
250 TRAP 195
270 OPEN #1,8,0,"P:"
280 REM GO AROUND PRINTER
300 GOSUB 1000:REM INTRODUCTION
310 GOSUB 2000:REM SETUP
320 GOSUB 3000:REM PLAY
330 GOSUB 4000:REM END ROUTINE
340 POKE 702,64:REM SETS UPPER CASE
350 END
1000 REM INTRODUCTION
1010 POKE 752,1:REM INVISIBLE CURSOR
1020 PRINT "}"
1030 PRINT "          "
1040 PRINT "          "
1060 PRINT "  Science-Fiction Tales   "
1065 PRINT "          "
1070 PRINT "        from          "
1075 PRINT "          "
```

```
1080 PRINT "     the Electronic Mind     "
1090 PRINT "               "
1100 PRINT "               "
1140 B=256
1150 IF A=B THEN 1200
1155 SOUND 0,B,10,10
1160 FOR I=1 TO 10:NEXT I
1165 SOUND 0,A,10,10
1170 A=A+1
1175 B=B-1
1180 GOTO 1150
1200 SOUND 0,0,0,0
1205 PRINT :PRINT
1210 E$="This program will produce endless funny little science fiction stories
     for your reading pleasure."
1230 F$="You may enter characters and places of your choice in the stories in
     order to make them personal."
1300 A=0
1310 IF A=LEN(E$) THEN 1400
1320 SOUND 0,A,10,10
1330 A=A+1
1340 PRINT E$(A,A);
1350 FOR I=1 TO 15:NEXT I
1360 GOTO 1310
1400 PRINT :PRINT
1410 B=1:A=A+1
1420 IF B=LEN(F$)+1 THEN SOUND 0,0,0,0:GOTO 1500
1430 SOUND 0,A,10,10
1440 PRINT F$(B,B);
1450 A=A-1:B=B+1
1460 FOR I=1 TO 10:NEXT I
1470 GOTO 1420
1500 PRINT :PRINT
1510 PRINT "Please log in your name";
1520 POKE 702,0:REM SETS LOWER CASE
1525 INPUT NA$
1530 IF NA$="" THEN NA$="Nameless Blob":PRINT "OK, we will call you ";NA$
1535 FOR I=1 TO 300:NEXT I
1540 PRINT "}":PRINT :PRINT
1550 PRINT "The stories can contain:"
1560 PRINT :PRINT
1570 PRINT "A. Your selected names."
1580 PRINT :PRINT "B. The programs list of names."
1590 PRINT :PRINT "C. Both lists of names."
```

```
1600 PRINT :PRINT "Press A,B or C:"
1610 INPUT G$
1620 IF G$="A" OR G$="a" OR G$="" THEN HG=2
1630 IF G$="B" OR G$="b" THEN HG=1:GOTO 220
1640 IF G$="C" OR G$="c" THEN HG=0
1990 RETURN
2000 REM SETUP
2010 POKE 752,0:REM VISIBLE CURSOR
2020 PRINT "}"
2025 DIM AA$(20),AB$(20),AC$(20),AD$(20),AE$(20)
2030 DIM BA$(20),BB$(20),BC$(20),BD$(20),BE$(20)
2040 DIM CA$(20),CB$(20),CC$(20),CD$(20),CE$(20)
2045 DIM DA$(20),DB$(20),DC$(20),DD$(20),DE$(20)
2050 PRINT "Enter five names of characters who will save the day. (20 letters max)."
2055 PRINT ""
2060 INPUT AA$,AB$,AC$,AD$,AE$
2070 PRINT "Enter five names of grizzly monsters who will attack."
2075 PRINT ""
2080 INPUT BA$,BB$,BC$,BD$,BE$
2082 PRINT "}"
2085 PRINT "Enter five names of places that will be attacked."
2090 PRINT ""
2095 INPUT CA$,CB$,CC$,CD$,CE$
2100 PRINT "Enter the names of five places from which monsters come."
2110 PRINT ""
2120 INPUT DA$,DB$,DC$,DD$,DE$
2125 POKE 752,1
2130 RETURN
2199 REM HEROS
2200 A$="Tim Conway"
2202 RETURN
2205 A$="Big Bird"
2207 RETURN
2210 A$="The Fonz"
2212 RETURN
2215 A$="Wonder Woman"
2217 RETURN
2220 A$="Kareem Abdul Jabar"
2222 RETURN
2225 A$="Tom Selleck"
2227 RETURN
2230 A$="Lucille Ball"
2232 RETURN
```

```
2235 A$=AA$
2237 RETURN
2240 A$=AB$
2242 RETURN
2245 A$=AC$
2247 RETURN
2250 A$=AD$
2252 RETURN
2255 A$=AE$
2257 RETURN
2270 A$=AC$
2299 REM NAMES OF MONSTERS
2300 A$="Transylvanian vampires"
2302 RETURN
2305 A$="Brazilian killer bees"
2307 RETURN
2310 A$="Giant green worms"
2312 RETURN
2315 A$="Mubbles"
2317 RETURN
2320 A$="bug eyed monsters"
2322 RETURN
2325 A$="rabid androids"
2327 RETURN
2330 A$="fetid zombies"
2332 RETURN
2335 A$=BA$
2337 RETURN
2340 A$=BB$
2342 RETURN
2345 A$=BC$
2347 RETURN
2350 A$=BD$
2352 RETURN
2355 A$=BE$
2357 RETURN
2399 REM PLACES THAT HEROS ARE FROM
2400 A$="Disneyland"
2402 RETURN
2405 A$="Cucumonga"
2407 RETURN
2410 A$="The United States"
2412 RETURN
2415 A$="Okeefenokee Swamp"
```

```
2417 RETURN
2420 A$="Tucumcari"
2422 RETURN
2425 A$="Mt. St. Helens"
2427 RETURN
2430 A$="Timbuktu"
2432 RETURN
2435 A$=CA$
2437 RETURN
2440 A$=CB$
2442 RETURN
2445 A$=CC$
2447 RETURN
2450 A$=CD$
2452 RETURN
2455 A$=CE$
2457 RETURN
2499 REM PLACES THAT MONSTERS ARE FROM
2500 A$="Planet XX"
2502 RETURN
2505 A$="Persepeon"
2507 RETURN
2510 A$="Bondegarra"
2512 RETURN
2515 A$="under the bed"
2517 RETURN
2520 A$="the mausoleum"
2522 RETURN
2525 A$="downtown Burbank"
2527 RETURN
2530 A$="The I.R.S."
2532 RETURN
2535 A$=DA$
2537 RETURN
2540 A$=DB$
2542 RETURN
2545 A$=DC$
2547 RETURN
2550 A$=DD$
2552 RETURN
2555 A$=DE$
2557 RETURN
3000 REM PLAY
```

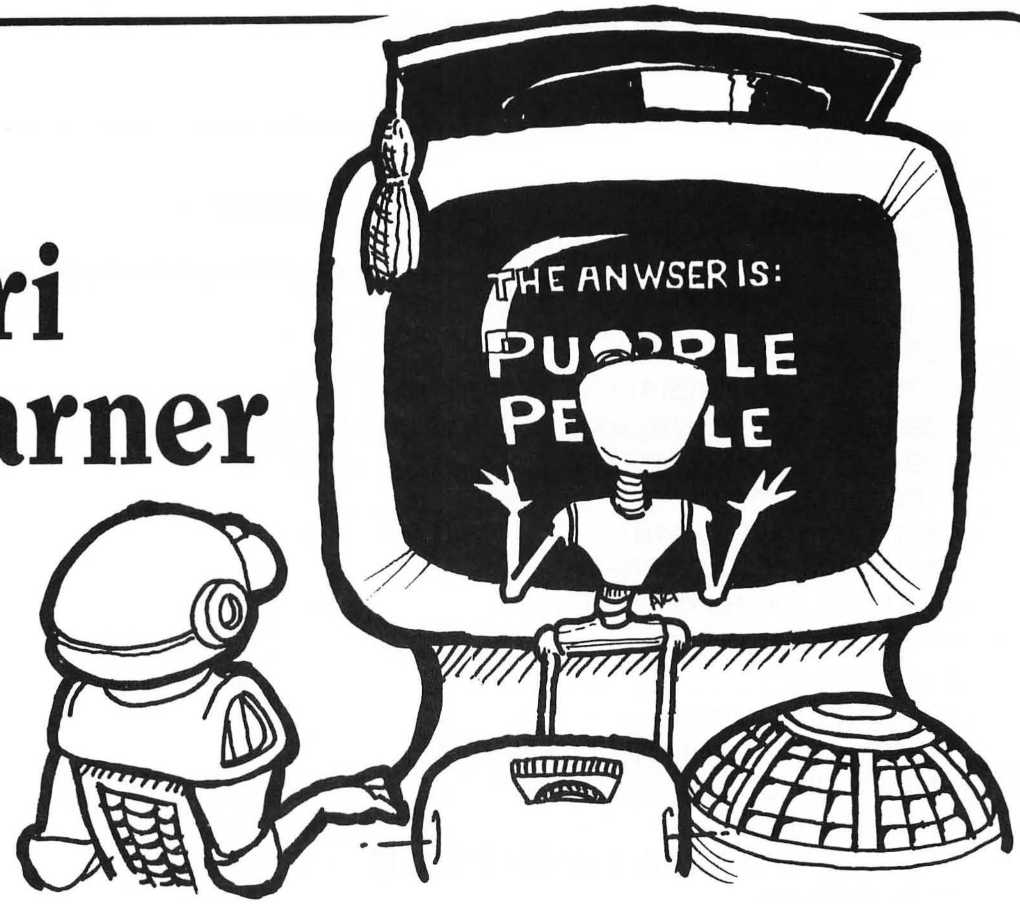
```
3ØØ5 SETCOLOR 2,5,Ø
3Ø1Ø R=Ø:S=Ø
3Ø2Ø PRINT "}"
32ØØ R=INT(RND(Ø)*5)+1
32Ø5 ON R GOTO 321Ø,3215,322Ø,3225,323Ø
321Ø PRINT "######### News Bulletin!!!#########":GOTO 325Ø
3215 PRINT "######### Flash! Flash! Flash!########":GOTO 325Ø
322Ø PRINT "######### Attention Earth!! #########":GOTO 325Ø
3225 PRINT "###### Calling All Earthlings!######":GOTO 325Ø
323Ø PRINT " ###### Now Hear This!!######"
325Ø PRINT :PRINT
3255 R=5*(INT(RND(Ø)*((12*(HG=Ø))+(7*(HG=))+(5*(HG=2))))+(7*(HG=2)))
326Ø GOSUB (24ØØ+R)
327Ø GOSUB 38ØØ
33ØØ R=INT(RND(Ø)*5)+1
33Ø5 ON R GOTO 331Ø,3315,332Ø,3325,333Ø
331Ø A$=" was attacked at dawn by ":GOSUB 38ØØ
3312 GOTO 335Ø
3315 A$=" has been surrounded and infiltrated by ":GOSUB 38ØØ
3317 GOTO 335Ø
332Ø A$=" appears to be under attack from ":GOSUB 38ØØ
3322 GOTO 335Ø
3325 A$=" has been whomped and stomped by ":GOSUB 38ØØ
3327 GOTO 335Ø
333Ø A$=" was rudely jarred awake by ":GOSUB 38ØØ
335Ø R=(INT(RND(Ø)*(7+S)))*5
336Ø GOSUB (23ØØ+R)
337Ø GOSUB 38ØØ
34ØØ R=INT(RND(Ø)*5)+1
34Ø5 ON R GOTO 341Ø,3415,342Ø,3425,343Ø
341Ø A$=" from the lowest levels of ":GOSUB 38ØØ
3412 GOTO 345Ø
3415 A$=" out of the vastness of ":GOSUB 38ØØ
3417 GOTO 345Ø
342Ø A$=" descended from ":GOSUB 38ØØ
3422 GOTO 345Ø
3425 A$=" from the greasy pits of ":GOSUB 38ØØ
3427 GOTO 345Ø
343Ø A$=" who oozed away from ":GOSUB 38ØØ
345Ø R=5*(INT(RND(Ø)*((12*(HG=Ø))+(7*(HG=1))+(5*(HG=2))))+(7*(HG=2)))
346Ø GOSUB (25ØØ+R)
347Ø GOSUB 38ØØ
35ØØ R=5*(INT(RND(Ø)*((12*(HG=Ø))+(7*(HG=1))+(5*(HG=2))))+(7*(HG=2)))
35Ø5 A$=". Our hero ":GOSUB 38ØØ
```

```
3510 GOSUB (2200+R)
3520 GOSUB 3800
3550 R=INT(RND(0)*5)+1
3555 ON R GOTO 3560,3565,3570,3575,3580
3560 A$=" dropped dozens of banana peels.":GOSUB 3800
3562 GOTO 3600
3565 A$="  tried to tie their shoelaces together. ":GOSUB 3800
3567 GOTO 3600
3570 A$=" told their mothers they were naughty.":GOSUB 3800
3572 GOTO 3600
3575 A$=" used a huge pea shooter against them.":GOSUB 3800
3577 GOTO 3600
3580 A$=" deployed tactical nuclear weapons.":GOSUB 3800
3600 R=INT(RND(0)*5)+1
3605 ON R GOTO 3610,3615,3620,3625,3630
3610 A$=" No use! They kept on coming just like a lava flow. ":GOSUB 3800
3612 GOTO 3650
3615 A$=" In spite of this heroism they were not deterred. ":GOSUB 3800
3617 GOTO 3650
3620 A$=" Nothing slowed them down. ":GOSUB 3800
3622 GOTO 3650
3625 A$=" However those baddies couldn't be stopped. ":GOSUB 3800
3627 GOTO 3650
3630 A$=" 'Ha Ha' they yelled and ran even faster. ":GOSUB 3800
3650 R=5*(INT(RND(0)*((12*(HG=0))+(7*(HG=1))+(5*(HG=2))))+(7*(HG=2)))
3660 GOSUB (2200+R)
3670 GOSUB 3800
3700 R=INT(RND(0)*5)+1
3705 ON R GOTO 3710,3715,3720,3725,3730
3710 A$=" held up a mojo and it was GAME OVER for the baddies.":GOSUB 3800
3712 GOTO 4080
3715 A$=" Spilled toxic waste in their path, so they ran off.":GOSUB 3800
3717 GOTO 4080
3720 A$=" hired them to make sci-fi movies.":GOSUB 3800
3722 GOTO 4080
3725 A$=" demanded their union cards so they left the set.":GOSUB 3800
3727 GOTO 4080
3730 A$=" held up a mirror and neutralized them.":GOSUB 3800
3735 GOTO 4080
3800 REM PRINT OUT AND PLAY MUSIC
3805 IF A$="" THEN A$=NA$
3810 PT=1:NT=0
3815 PRINT A$(PT,PT);
```

```
3820 READ NT,TM
3825 SOUND 0,NT,10,10
3830 IF PEEK(85)>30 AND A$(PT,PT)=" " THEN PRINT
3835 IF NT=0 THEN RESTORE
3840 PT=PT+1
3845 FOR I=1 TO (TM*2):NEXT I
3850 SOUND 0,0,0,0
3855 IF PT<=LEN(A$) THEN GOTO 3815
3860 RETURN
3900 REM MUSIC NOTES HERE
3910 DATA 128,1,128,1,128,2,128,1,128,1,128,2,128,1,128,1,96,2,85,2,
     76,2,128,1,128,1,128,2,128,1,128,1
3920 DATA 96,2,76,1,76,1,85,2,102,2,128,2,128,1,128,1,128,2,128,1,128,1,
     128,2,128,1,128,1,96,2,85,2,76,2,96,1
3925 DATA 76,1,64,4,64,1,72,1,76,1,85,1,96,2,76,2,96,2,128,1,128,1,128,2,
     128,1,128,1,128,2,128,1,128,1
3930 DATA 96,2,85,2,76,2,128,1,128,1,128,2,128,1,128,1,96,2,76,1,76,1,85,
     2,102,2,128,2,128,1,128,1,128,2
3935 DATA 128,1,128,1,128,2,128,1,128,1,96,2,85,2,76,2,96,1,76,1,64,4
3940 DATA 64,1,72,1,76,1,85,1,96,2,76,2,96,2,76,1,76,1,76,2,76,1,76,1,76,2,
     76,1,76,1,76,2,57,2,76,2,57,2,76,2
3950 DATA 57,2,76,2,85,2,96,2,102,2,114,2,76,1,76,1,76,2,76,1,76,1,76,2,76,
     1,76,1,76,2,57,2,76,2,57,2
3960 DATA 76,2,57,2,64,2,68,2,64,2,68,2,64,2,76,1,76,1,76,2,76,1,76,1,76,2,
     76,1,76,1,76,2,57,2,76,2,57,2
3970 DATA 76,2,57,2,76,2,85,2,96,2,102,2,114,2,76,1,76,1,76,2,76,1,76,1,76,
     2,76,1,76,1,76,2,57,2,76,2,57,2
3980 DATA 76,2,57,2,64,2,68,2,64,6
3995 DATA 0,0
4000 REM END ROUTINE
4010 PRINT "}"
4020 PRINT "############ADIOS###############"
4030 RETURN
4080 SOUND 0,0,0,0
4090 PRINT :PRINT
4100 PRINT "Had enough of our little yarns?"
4110 PRINT "Press 'p' to print the story"
4115 PRINT "Press 'x' to exit,[RETURN] to go on."
4120 INPUT G$
4130 IF G$="" THEN G$="NO"
4140 IF G$(1,1)="P" OR G$(1,1)="p" THEN GOTO 4200
4170 IF G$="" THEN G$="NO"
4180 IF G$(1,1)="X" OR G$(1,1)="x" THEN PRINT "}";:RETURN
4190 GOTO 3000
4200 A1=PEEK(561)*256+PEEK(560)
```

```
4210 A2=PEEK(A1+5)*256+PEEK(A1+4)
4220 FOR X=0 TO 12
4222 REM TO PRINT A FULL SCREEN CHANGE 12 TO 23
4230 FOR Y=0 TO 39
4240 KS=PEEK(A2):IF KS<97 THEN KS=KS+32
4245 PRINT #1;CHR$(KS);
4250 A2=A2+1
4260 NEXT Y:PRINT #1
4270 NEXT X
4300 PRINT "}"
4310 GOTO 3000
4990 RETURN
```

Atari Learner



Your Atari Home Computer can learn to play a guessing game with you. Like all computers, your Atari starts out with an open mind. Atari Learner is a type of data base program that will accumulate data until it knows 50 objects and can tell the difference between them by asking appropriate questions. Atari BASIC does not have built-in string arrays. That means you cannot say A$(1)="cat" and A$(2)="banana." That would be too easy. We have to set up pseudo arrays by multiplying the number of elements by a fixed length number of characters in each element. This is done in the DIM statements at 1100 to 1130. The number of elements is set to 50 in line 1060 and assigned to the variable MX. In line 1100 the string that will hold the questions [QU$] is multiplied by MX times 40 +1. The extra 1 is there because there is no 0th array and the counter must start with one. The program does not test for string length and throw away extra spaces because each word and question is printed on its own line and the extra blank spaces do not show. The file is a straight sequential file, but the elements are of fixed length like a random access file. You could use a program like this to write a menu planner or other data base. If you have a disk drive be sure to initialize an empty disk first for saving files. You will not be able to initialize a disk while a program is running.

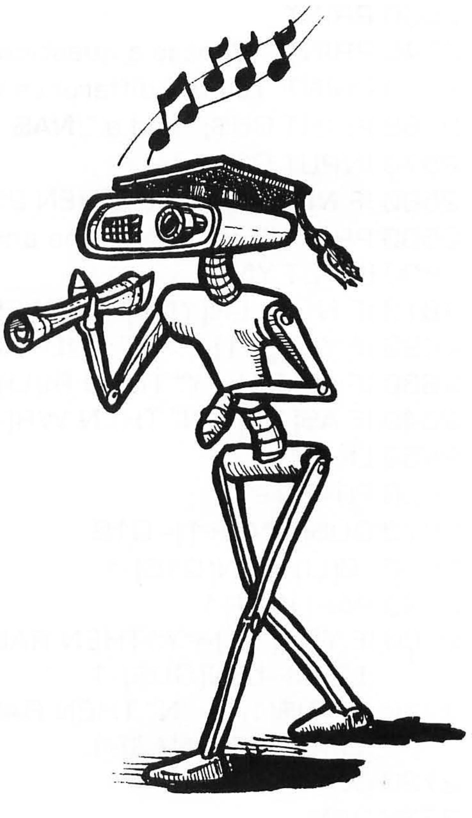
```
1000 REM *****************************************************************
1010 REM *                                                               *
1020 REM *                        ATARI LEARNER                          *
1030 REM *                                                               *
1040 REM *****************************************************************
1050 REM
1060 MX=50
1070 DIM A$(4),TA$(4),YN$(4),GU$(40)
1080 DIM RI(MX),WR(MX),FN$(20),Q1$(41)
1090 DIM L0(MX),L1(MX),L2(MX)
1100 DIM QU$(MX*40+1)
1110 DIM RA$(MX*40+1)
1120 DIM WA$(MX*40+1)
1130 DIM NA$(MX*40+1)
1140 GOTO 1260
1150 PRINT "}";
1160 POSITION 10,4
1170 PRINT "*** ATARI LEARNER ***"
1180 POSITION 2,8
1190 RETURN
1200 Q1$=QU$(LI*40+1,LI*40+1+L0(LI))
1210 RETURN
1220 GU$=RA$(LI*40+1,LI*40+1+L1(LI))
1230 RETURN
1240 GU$=WA$(LI*40+1,LI*40+1+L2(LI))
1250 RETURN
1260 GRAPHICS 0
1270 GOSUB 1150
1280 PRINT "This is a game that has the ability"
1290 PRINT "to learn. It will attempt to guess"
1300 PRINT "the name of an object that you pick"
1310 PRINT "at random."
1320 PRINT
1330 PRINT "Whenever you stump the computer, you"
1340 PRINT "are asked about the object you"
1350 PRINT "selected. By compiling this"
1360 PRINT "information, the computer learns."
1370 PRINT
1380 PRINT
1390 PRINT "enter 'STOP' when you are done."
1400 POSITION 2,23
1410 PRINT "Press 'RETURN' when ready to play";
1420 REM
1430 QU$(41)="Does it move along the ground"
```

```
1440 LØ(1)=28
1450 RA$(41)="Car"
1460 L1(1)=2
1470 WA$(41)="House"
1480 L2(1)=4
1490 RI(1)=Ø
1500 WR(1)=Ø
1510 FR=2
1520 FOR LP=2 TO MX
1530 LØ(LP)=Ø
1540 L1(LP)=Ø
1550 L2(LP)=Ø
1560 RI(LP)=Ø
1570 WR(LP)=Ø
1580 NEXT LP
1590 INPUT A$
1600 REM
1610 PRINT "}"
1620 POSITION 2,4
1630 PRINT "Would you like to load a previously"
1640 PRINT "saved file of data Y/N ";
1650 INPUT A$
1660 IF LEN(A$) THEN IF A$(1,1)< >"Y" THEN 2030
1670 PRINT
1680 PRINT
1690 PRINT "The devices are :"
1700 PRINT " C: for cassette"
1710 PRINT " D: for diskette"
1720 PRINT
1730 PRINT "Enter :"
1740 PRINT " Device:Filename.Ext ";
1750 INPUT FN$
1760 IF LEN(FN$)<3 THEN 1610
1770 OPEN #1,4,Ø,FN$
1780 INPUT #1;FR
1790 GET #1,I
1800 FOR L=1 TO I
1810 GET #1,A
1820 RA$(L,L)=CHR$(A)
1830 NEXT L
1840 GET #1,I
1850 FOR L=1 TO I
1860 GET #1,A
1870 WA$(L,L)=CHR$(A)
```

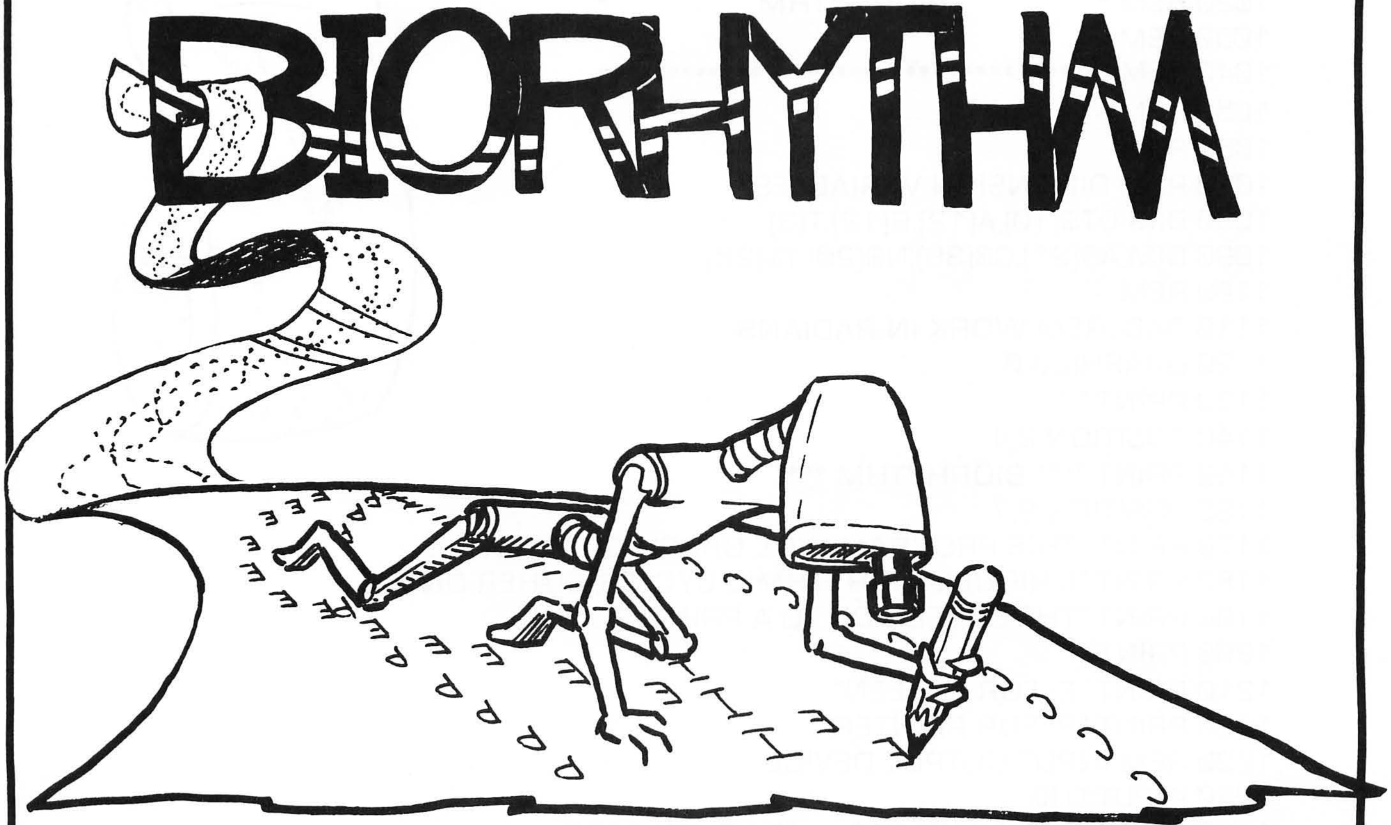
```
1880 NEXT L
1890 GET #1,I
1900 FOR L=1 TO I
1910 GET #1,A
1920 QU$(L,L)=CHR$(A)
1930 NEXT L
1940 FOR LP=1 TO FR
1950 INPUT #1;A:RI(LP)=A
1960 INPUT #1;A:WR(LP)=A
1970 INPUT #1;A:L0(LP)=A
1980 INPUT #1;A:L1(LP)=A
1990 INPUT #1;A:L2(LP)=A
2000 NEXT LP
2010 CLOSE #1
2020 REM
2030 REM PLAY THE GAME
2040 REM
2050 LI=1
2060 GOSUB 1150
2070 PRINT "I know of ";FR;" objects..."
2080 PRINT
2090 GOSUB 1200:REM GET Q1$
2100 PRINT Q1$;" ";
2110 INPUT A$
2120 IF NOT LEN(A$) THEN 2100
2130 IF A$(1,1)="Y" THEN 2190
2140 IF A$(1,1)="N" THEN 2220
2150 IF A$(1,1)="S" THEN 2740
2160 PRINT "Please answer Yes or No"
2170 PRINT
2180 GOTO 2100
2190 IF RI(LI) THEN LI=RI(LI):GOTO 2090
2200 GOSUB 1220:REM GET GU$ FROM RA$
2210 GOTO 2240
2220 IF WR(LI) THEN LI=WR(LI):GOTO 2090
2230 GOSUB 1240:REM GET GU$ FROM WA$
2240 PRINT "Is it a ";GU$;" ";
2250 INPUT TA$
2260 IF NOT LEN(TA$) THEN 2390
2270 IF TA$(1,1)<>"Y" THEN 2390
2280 PRINT
2290 PRINT
2300 PRINT "I GOT IT !!!"
2310 Y=-100
```

```
2320 FOR PA=1 TO 400
2330 SOUND 0,ABS(Y),8,5
2340 Y=Y+2
2350 IF Y>100 THEN Y=-100
2360 NEXT PA
2370 SOUND 0,0,0,0
2380 GOTO 2050
2390 IF TA$(1,1)="S" THEN 2740
2400 PRINT
2410 PRINT
2420 PRINT
2430 PRINT "What is the object ";
2440 INPUT NA$
2450 IF FR<=MX THEN 2530
2460 PRINT "Sorry I can't remember that one."
2470 PRINT "My memory seems to be full"
2480 SOUND 0,130,12,5
2490 FOR PA=1 TO 300
2500 NEXT PA
2510 SOUND 0,0,0,0
2520 GOTO 2050
2530 PRINT
2540 PRINT "What is a question I would use to"
2550 PRINT "tell the difference between a"
2560 PRINT GU$;" and a ";NA$
2570 INPUT Q1$
2580 IF NOT LEN(Q1$) THEN 2530
2590 PRINT "For ";NA$;" the answer is what Y/N ";
2600 INPUT YN$
2610 IF NOT LEN(YN$) THEN 2590
2620 IF YN$(1,1)< >"Y" AND YN$(1,1)< >"N" THEN 2590
2630 IF A$(1,1)="Y" THEN RI(LI)=FR
2640 IF A$(1,1)="N" THEN WR(LI)=FR
2650 LI=FR
2660 FR=FR+1
2670 QU$(LI*40+1)=Q1$
2680 L0(LI)=LEN(Q1$)-1
2690 PA=LI*40+1
2700 IF YN$(1,1)="Y" THEN RA$(PA)=NA$:L1(LI)=LEN(NA$)-1:WA$(PA)=GU$
     :L2(LI)=LEN(GU$)-1
2710 IF YN$(1,1)="N" THEN RA$(PA)=GU$:L1(LI)=LEN(GU$)-1:WA$(PA)=NA$
     :L2(LI)=LEN(NA$)-1
2720 GOTO 2050
2730 REM
```

```
2740 GOSUB 1150
2750 PRINT "Would you like to save the file Y/N";
2760 INPUT A$
2770 IF NOT LEN(A$) THEN 2820
2780 IF A$(1,1)="N" THEN 2820
2790 GOSUB 1150
2800 PRINT "Last file used was ";FN$
2810 PRINT
2820 PRINT "The devices are :"
2830 PRINT " C: for cassette"
2840 PRINT " D: for diskette"
2850 PRINT
2860 PRINT "Enter :"
2870 PRINT " Device:Filename.Ext ";
2880 INPUT FN$
2890 IF LEN(FN$)<3 THEN 2740
2900 OPEN #1,8,0,FN$
2910 PRINT #1;FR
2920 PUT #1,LEN(RA$)
2930 FOR L=1 TO LEN(RA$)
2940 PUT #1,ASC(RA$(L,L))
2950 NEXT L
2960 PUT #1,LEN(WA$)
2970 FOR L=1 TO LEN(WA$)
2980 PUT #1,ASC(WA$(L,L))
2990 NEXT L
3000 PUT #1,LEN(QU$)
3010 FOR L=1 TO LEN(QU$)
3020 PUT #1,ASC(QU$(L,L))
3030 NEXT L
3040 FOR LP=1 TO FR
3050 PRINT #1;RI(LP)
3060 PRINT #1;WR(LP)
3070 PRINT #1;L0(LP)
3080 PRINT #1;L1(LP)
3090 PRINT #1;L2(LP)
3100 NEXT LP
3110 CLOSE #1
3120 END
```

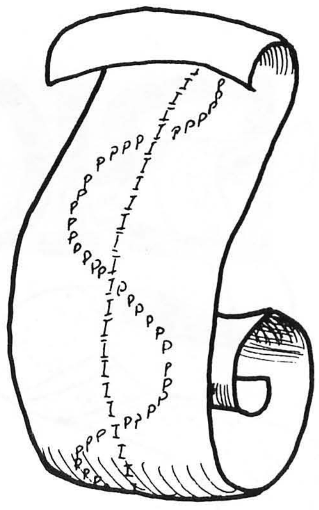


BIORHYTHM

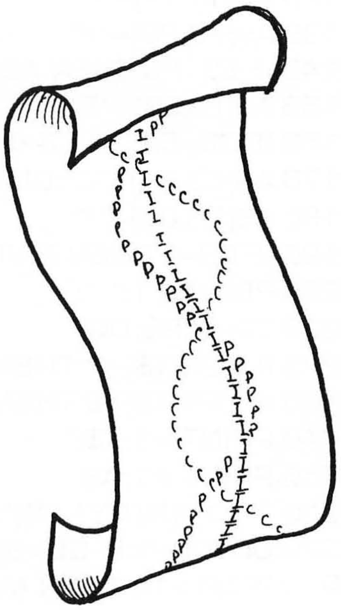


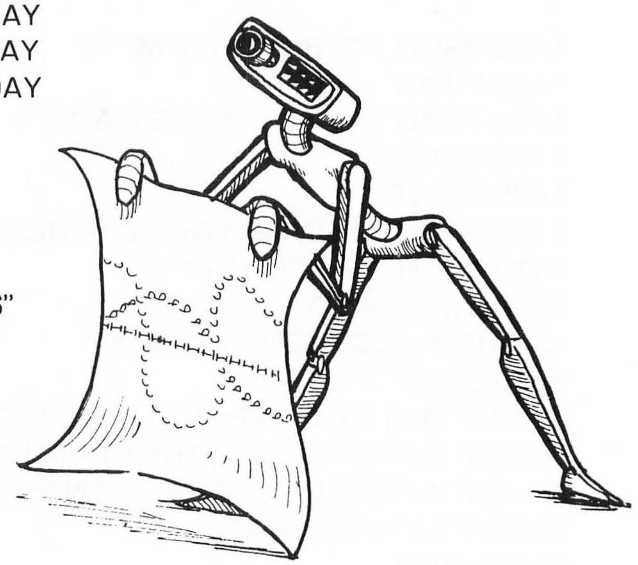
Do you believe in biorhythms? How about Santa Claus, or the Tooth Fairy? We do not take an official position on the veracity of any of these worthwhile institutions. What is offered in this program is an example of an excellent date formating routine that could form the basis of a mortgage amortization program or other heavy duty calendar based calculator. The beautiful curved lines of the biorhythm printout are generated by the built-in sine wave functions of Atari BASIC. You can demonstrate the SIN function by putting line 2060 into the "basement workshop" that is described in the introduction. Biorhythms is the sample program used in the introduction to expain how to tinker with variables. The print statements are all PRINT #1, which means print to a device. In the beginning of the program you are asked to chose to print to the screen (E:) or to a printer (P:). Lines 2090 to 2110 fills a string (A$) with 21 blank spaces. Line 2120 puts a straight line in the middle of the string (11,11). Lines 2130 to 2200 figures the proper position in A$ for the physical (P), cognitive (C), and emotional (E) to be printed. If one letter comes on top of another letter a "*" is printed for the conjunction of the two cycles. The remaining print statements carefully space the month and day calculations into neat columns.

```
1000 REM  ***********************************
1010 REM *                                  *
1020 REM *          BIORHYTHM               *
1030 REM *                                  *
1040 REM ***********************************
1050 REM
1060 REM
1070 REM DIMENSION VARIABLES
1080 DIM OT$(10),A(12),B(12),T(3)
1090 DIM A$(21),C$(36),N$(20),T$(20)
1100 REM
1110 RAD :REM WORK IN RADIANS
1120 GRAPHICS 0
1130 PRINT "}"
1140 POSITION 2,4
1150 PRINT "*** BIORHYTHM ***"
1160 POSITION 2,7
1170 PRINT "THIS PROGRAM WILL GRAPH OUT YOUR"
1180 PRINT "UNIQUE BIORHYTHMIC CYCLES, EITHER ON"
1190 PRINT "THE SCREEN OR TO A PRINTER"
1200 PRINT
1210 PRINT "E: FOR SCREEN"
1220 PRINT "P: FOR PRINTER"
1230 REM INPUT OUTPUT DEVICE
1240 INPUT OT$
1250 IF OT$< >"E:" AND OT$< >"P:" THEN 1130
1260 REM
1270 REM SETUP
1280 REM
1290 RESTORE
1300 C$="JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
1310 FOR I=1 TO 12
1320 READ A:A(I)=A
1330 NEXT I
1340 FOR I=1 TO 12
1350 READ A:B(I)=A
1360 NEXT I
1370 DATA 0,31,59,90,120,151,181,212,243,273,304,334
1380 DATA 31,28,31,30,31,30,31,31,30,31,30,31
1390 REM
1400 REM ** PLAY GAME
1410 REM
1420 PRINT "}"
1430 POSITION 12,4
```



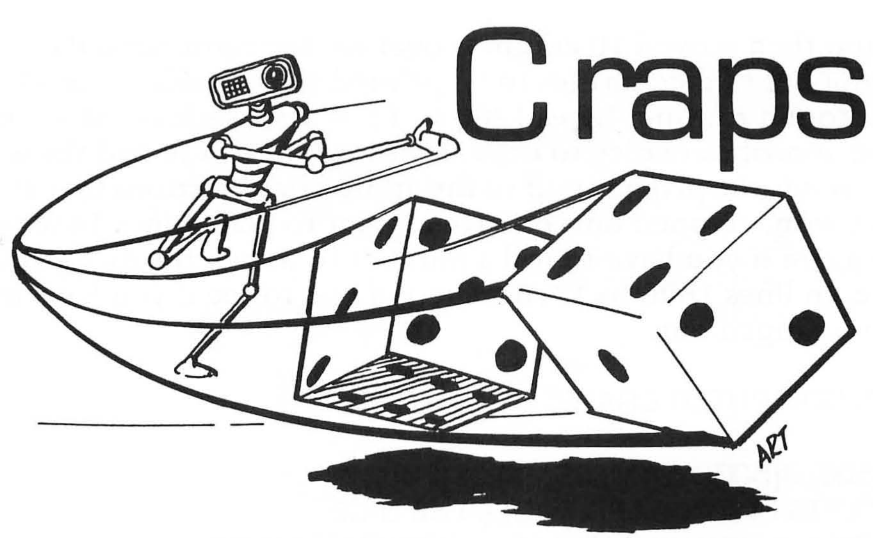
```
1440 PRINT "*** BIORHYTHM ***"
1450 PRINT
1460 PRINT "WHAT IS YOUR NAME. ";
1470 INPUT N$
1480 PRINT
1490 PRINT "WHAT IS YOUR BIRTHDATE."
1500 PRINT "MM,DD,YYYY ";
1510 INPUT M,D,Y
1520 IF Y<100 THEN Y=Y+1900
1530 ER=0
1540 IF M<1 OR M>12 THEN PRINT "INCORRECT MONTH":ER=1
1550 IF D<1 OR D>B(M) THEN PRINT "INCORRECT DAY":ER=1
1560 IF Y<1900 OR Y>2100 THEN PRINT "INCORRECT YEAR ":ER=1
1570 IF ER THEN 1490
1580 PRINT
1590 PRINT "WHAT IS THE CHART START DATE"
1600 PRINT "MM,DD,YYYY ";
1610 INPUT M1,D0,Y1
1620 IF Y1<100 THEN Y1=Y1+1900
1630 ER=0
1640 IF M1<1 OR M1>12 THEN PRINT "INCORRECT MONTH":ER=1
1650 IF D0<1 OR D0>B(M1) THEN PRINT "INCORRECT DAY":ER=1
1660 IF Y1<1900 OR Y1>2100 THEN PRINT "INCORRECT YEAR ":ER=1
1670 IF ER THEN 1610
1680 PRINT
1690 PRINT "HOW MANY DAYS ";
1700 INPUT Z
1710 IF Z<1 OR Z< >INT(Z) THEN 1680
1720 W=D0
1730 W1=M1
1740 W2=Y1
1750 W3=Z
1760 J=A(M)+D
1770 D1=365-J+((J<=60) AND (Y/4=INT(Y/4)))
1780 D2=365*(Y1-(Y+1))
1790 E=0
1800 FOR T=Y+1 TO Y1-1
1810 E=E+(T/4=INT(T/4))
1820 NEXT T
1830 D3=A(M1)+D0
1840 D3=D3+((Y/4=INT(Y/4)) AND (D3>=60))
1850 D4=D1+D2+D3+E
1860 IF D4<0 THEN PRINT "START IS BEFORE BIRTHDATE":GOTO 1590
```





```
1870 P1=D4-INT(D4/23)*23:REM 23 DAY
1880 E1=D4-INT(D4/28)*28:REM 28 DAY
1890 C1=D4-INT(D4/33)*33:REM 33 DAY
1900 PRINT "}"
1910 PRINT "PRINTING CHART"
1920 PRINT
1930 CLOSE #1
1940 OPEN #1,8,Ø,OT$
1950 PRINT #1;"  BIORHYTHM CYCLES"
1960 PRINT #1;" ----- FOR ----- "
1970 PRINT #1;"  ";N$
1980 PRINT #1;C$(3*M-2,3*M);" ";D;
1990 PRINT #1;", ";Y
2000 PRINT #1;C$(3*M1-2,3*M1);" ";
2010 PRINT #1;Y1;"          (-) (Ø) (+)"
2020 FOR T=1 TO Z
2030 P2=(P1+T)-INT((P1+T)/23)*23
2040 E2=(E1+T)-INT((E1+T)/28)*28
2050 C2=(C1+T)-INT((C1+T)/33)*33
2060 P3=INT(11.5+10*SIN(P2*6.283/23))
2070 E3=INT(11.5+10*SIN(E2*6.283/28))
2080 C3=INT(11.5+10*SIN(C2*6.283/33))
2090 FOR I=1 TO 21
2100 A$(I,I)=" "
2110 NEXT I
2120 A$(11,11)=":"
2130 A$(P3,P3)="P"
2140 IF E3=P3 THEN A$(E3,E3)="*":GOTO 2160
2150 A$(E3,E3)="E"
2160 IF C3=E3 OR C3=P3 THEN 2180
2170 A$(C3,C3)="C":GOTO 2190
2180 A$(C3,C3)="*"
2190 IF DØ=1 THEN PRINT #1;C$(3*M1-2,3*M1);" ";:GOTO 2210
2200 PRINT #1;"  ";
2210 T$=STR$(DØ)
2220 IF LEN(T$)=1 THEN PRINT #1;"  ";
2230 IF LEN(T$)=2 THEN PRINT #1;" ";
2240 PRINT #1;T$;"          ";
2250 PRINT #1;A$
2260 IF Y1-((INT(Y1/4))*4)=Ø THEN B(2)=29
2270 DØ=DØ+1:IF DØ>B(M1) THEN DØ=1:M1=M1+1
2280 IF M1>12 THEN M1=1:Y1=Y1+1
2290 NEXT T
2300 IF OT$="E:" THEN FOR PA=1 TO 1ØØØ:NEXT PA
2310 GOTO 113Ø
```

Craps



Here at last is a complete simulation of Las Vegas, including touting, insults, and the certainty of ending up broke! "Just double-up brother, your luck is bound to change!" One problem about doubling your bet every time you lose, is if you get a run of bad luck the Bank of England couldn't save you. The dice aren't really loaded as the screen suggests. You can see for yourself that the dice are "rolled" by the random number statements in lines 1150 and 1160. You can test your 'dice' with a routine that adds up occurrences of each number. You will find that certain numbers may be favored, but when you run the test again a different group of numbers will be favored, just like real life!

```
100 X=INT(6*RND(0)+1)
110 IF X=1 THEN A=A+1
120 IF X=2 THEN B=B+1
130 IF X=3 THEN C=C+1
140 IF X=4 THEN D=D+1
150 IF X=5 THEN E=E+1
160 IF X=6 THEN F=F+1
170 POSITION 2,5
180 PRINT A,B,C
190 PRINT
200 PRINT D,E,F
210 GOTO 100
```

In our version of craps a series of GOSUBs branch to the appropriate section where the dice are drawn in character graphics. The dice will appear on your screen as you type in the correct control characters, but they will not appear on a listing sent to a printer. Line 1260 is a POKE that sets the left hand margin to line up the printing of the dice. The margin is first read by a PEEK and then moved 10 columns over for the right-hand die. This allows the same set of dice images to be printed side-by-side. Line 1190 is a GOSUB to a sound routine. Lines 1200 to 1350 are a series of IF GOSUBs the select the appropriate dice to draw. Lines 1400 to 1430 add the dice together and sends the program off to the appropriate sections that announce that you have won, crapped out, or have a point to play. Lines 1430 to 1710 roll the dice again if you have to roll a number to match the dice you rolled the first time. In lines 1690 to 1710 there is a test to see if you have made your point or crapped out.

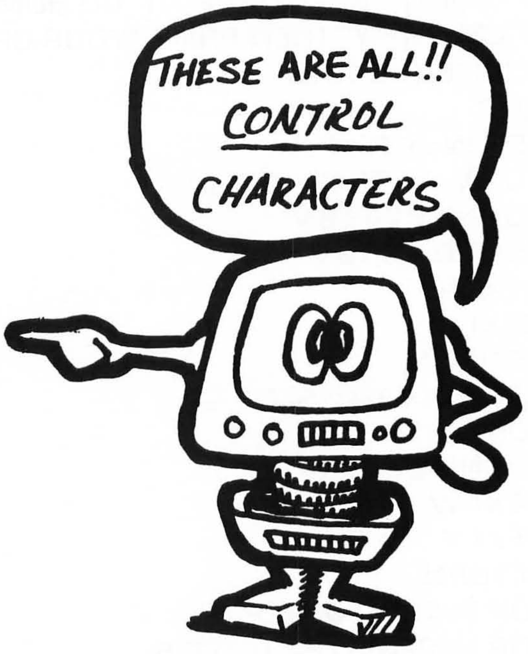
```
1000 PRINT "} COMPUTER CRAPS"
1010 S=200
1020 DIM S(500),B(500),A$(20),B$(20),C$(20)
1030 PRINT "LETS PLAY CRAPS. I ROLL THE DICE."
1040 PRINT "YOU KNOW THE RULES----YOU CAN'T WIN"
1050 PRINT " YOU HAVE $200 TO START"
1060 PRINT "PRESS 'RETURN' TO START THE GAME";
1070 INPUT B$
1080 IF S<=0 THEN 2350
1090 IF B>S THEN 2350
1100 IF B<51 THEN 1150
1110 PRINT "}"
1120 PRINT "PLACE AN AUTHORIZED BET,PLEASE! TABLE LIMIT IS $50."
1130 INPUT B
1140 GOTO 1100
1150 LET D1=INT(6*RND(0)+1)
1160 LET D2=INT(6*RND(0)+1)
1170 PRINT "}"
1180 D=D1+D2
1190 GOSUB 2400
1200 IF D1=1 THEN GOSUB 2610
1210 IF D1=2 THEN GOSUB 2690
1220 IF D1=3 THEN GOSUB 2770
1230 IF D1=4 THEN GOSUB 2850
1240 IF D1=5 THEN GOSUB 2930
1250 IF D1=6 THEN GOSUB 3010
1260 MAR=PEEK(82)
1270 POKE 82,MAR+10
1280 POSITION MAR+10,1
1290 IF D2=1 THEN GOSUB 2610
1300 IF D2=2 THEN GOSUB 2690
1310 IF D2=3 THEN GOSUB 2770
1320 IF D2=4 THEN GOSUB 2850
1330 IF D2=5 THEN GOSUB 2930
```

```
1340 IF D2=6 THEN GOSUB 3010
1350 POKE 82,MAR
1360 PRINT
1370 PRINT "YOU ROLLED A ";D1;" AND A ";D2;" WHICH IS ";D
1380 PRINT
1390 PRINT
1400 IF D1+D2<4 THEN 1970
1410 IF D1+D2=12 THEN 1970
1420 IF D1+D2=7 THEN 1730
1430 IF D1+D2=11 THEN 1820
1440 D1=INT(6*RND(0)+1)
1450 D2=INT(6*RND(0)+1)
1460 GOSUB 2400
1470 PRINT "}"
1480 IF D1=1 THEN GOSUB 2610
1490 IF D1=2 THEN GOSUB 2690
1500 IF D1=3 THEN GOSUB 2770
1510 IF D1=4 THEN GOSUB 2850
1520 IF D1=5 THEN GOSUB 2930
1530 IF D1=6 THEN GOSUB 3010
1540 MAR=PEEK(82)
1550 POKE 82,MAR+10
1560 POSITION MAR+10,1
1570 IF D2=1 THEN GOSUB 2610
1580 IF D2=2 THEN GOSUB 2690
1590 IF D2=3 THEN GOSUB 2770
1600 IF D2=4 THEN GOSUB 2850
1610 IF D2=5 THEN GOSUB 2930
1620 IF D2=6 THEN GOSUB 3010
1630 POKE 82,MAR
1640 PRINT
1650 PRINT "YOU ROLLED A ";D1;" AND ";D2;
1660 PRINT "                    "
1670 PRINT "YOU WANT A ";D;" THE POINT"
1680 PRINT
1690 IF D1+D2=7 THEN 2100
1700 IF D1+D2=D THEN 2200
1710 IF D1+D2< >D THEN 1440
1720 PRINT "YOU ROLLED A ";D1;" AND A ";D2;" WHICH IS ";D
1730 GOSUB 2470
1740 S=S+B
1750 PRINT "A LUCKY SEVEN! YOU WIN! YOU NOW HAVE $";S;"
     HOW MUCH DO YOU BET ";
1760 INPUT B
1770 PRINT
1780 PRINT
```

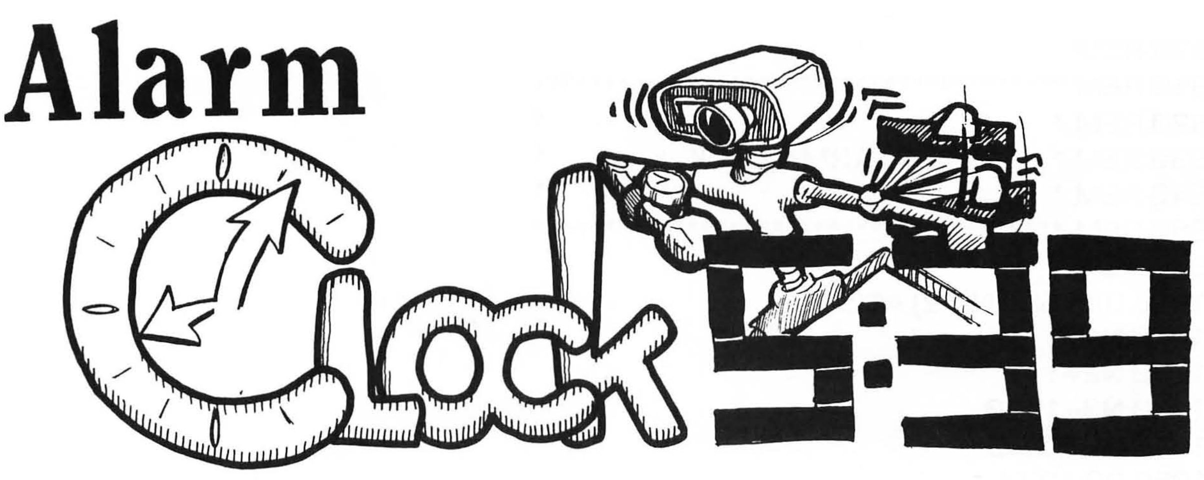
```
1790 PRINT "PRESS RETURN TO ROLL THE BONES AGAIN";
1800 INPUT B$
1810 GOTO 1080
1820 GOSUB 2470
1830 PRINT "YOU ROLLED A NATURAL WINNER!"
1840 PRINT
1850 PRINT "PIT BOSS,SOME NEW DICE"
1860 PRINT
1870 PRINT
1880 PRINT
1890 S=S+B
1900 PRINT "YOU NOW HAVE $";S;" HOW MUCH DO YOU BET"
1910 INPUT B
1920 PRINT
1930 PRINT
1940 PRINT "PRESS RETURN TO CONTINUE,SHOOTER";
1950 INPUT A$
1960 GOTO 1080
1970 GOSUB 2530
1980 PRINT
1990 PRINT "YOU ROLLED A CRAPS -- THE NAME OF THE GAME -- CHUCK
      IT IN-I KNOW YOU'LL WIN "
2000 S=S-B
2010 PRINT
2020 PRINT
2030 PRINT "YOU NOW HAVE $";S;" PLEASE PLACE YOUR BET";
2040 INPUT B
2050 PRINT
2060 PRINT
2070 PRINT "PRESS RETURN TO CONTINUE,SHOOTER";
2080 INPUT B$
2090 GOTO 1080
2100 GOSUB 2530
2110 PRINT "YOU LOSE BECAUSE YOU ROLLED A 7 - A
      LOSER -- SORRY -- DOUBLE UP AND CATCH UP"
2120 S=S-B
2130 PRINT "YOUR BANKROLL HAS DWINDLED TO $";S
2140 PRINT "DOUBLE YOUR BET NOW. I'LL SLIP YOU"
2150 PRINT "SOME CROOKED DICE.(THIS PROGRAM IS RIGGED)
      PLACE BET NOW";
2160 INPUT B
2170 PRINT "PRESS RETURN TO CONTINUE,SHOOTER";
2180 INPUT C$
2190 GOTO 1080
2200 GOSUB 2470
2210 PRINT
```

```
2220 PRINT " YOU ROLLED A ";D1;" AND A ";D2
2230 PRINT " WHICH IS ";D;"  THE POINT--A WINNER!"
2240 PRINT
2250 PRINT
2260 S=S+B
2270 PRINT "YOUR BANKROLL NOW IS $";S;" MAKE YOUR LUCKY BET";
2280 INPUT B$
2290 B=VAL(B$)
2300 PRINT
2310 PRINT
2320 PRINT "PRESS RETURN -- YOU'RE HOT"
2330 INPUT C$
2340 GOTO 1080
2350 PRINT "CAN YOU GET MORE MONEY";
2360 INPUT C$
2370 IF C$(1,1)="N" THEN PRINT "GO HOME,SLOB":END
2380 IF C$(1,1)="Y" THEN PRINT "YOUR CREDIT IS GOOD HERE FOR $500"
     :CLR :GOTO 1010
2390 END
2400 FOR W=1 TO 8
2410 FOR V=14 TO 0 STEP -2
2420 SOUND 0,243,6,V
2430 SOUND 1,8,10,V
2440 NEXT V
2450 NEXT W
2460 RETURN
2470 FOR V=14 TO 0 STEP -2
2480 FOR W=1 TO 25
2490 SOUND 0,60,10,V
2500 NEXT W
2510 NEXT V
2520 RETURN
2530 FOR V=15 TO 0 STEP -5
2540 FOR N=220 TO 240 STEP 5
2550 FOR W=1 TO 10
2560 SOUND 0,N,10,V
2570 NEXT W
2580 NEXT N
2590 NEXT V
2600 RETURN
2610 PRINT "[N][N][N][N][N]"
2620 PRINT "[V]             [B]"
2630 PRINT "[V]             [B]"
2640 PRINT "[V]      [T]    [B]"
```

```
2650 PRINT "[V]         [B]"
2660 PRINT "[V]         [B]"
2670 PRINT "[M][M][M][M][M]"
2680 RETURN
2690 PRINT "[N][N][N][N][N]"
2700 PRINT "[V]         [B]"
2710 PRINT "[V][T]      [B]"
2720 PRINT "[V]         [B]"
2730 PRINT "[V]      [T][B]"
2740 PRINT "[V]         [B]
2750 PRINT "[M][M][M][M][M]"
2760 RETURN
2770 PRINT "[N][N][N][N][N]"
2780 PRINT "[V]         [B]"
2790 PRINT "[V]      [T][B]"
2800 PRINT "[V]   [T]   [B]"
2810 PRINT "[V][T]      [B]"
2820 PRINT "[V]         [B]"
2830 PRINT "[M][M][M][M][M]"
2840 RETURN
2850 PRINT "[N][N][N][N][N]"
2860 PRINT "[V]         [B]"
2870 PRINT "[V][T]   [T][B]"
2880 PRINT "[V]         [B]"
2890 PRINT "[V][T]   [T][B]"
2900 PRINT "[V]         [B]"
2910 PRINT "[M][M][M][M][M]"
2920 RETURN
2930 PRINT "[N][N][N][N][N]"
2940 PRINT "[V]         [B]"
2950 PRINT "[V][T]   [T][B]"
2960 PRINT "[V]  [T]  [B]"
2970 PRINT "[V][T]   [T][B]"
2980 PRINT "[V]         [B]"
2990 PRINT "[M][M][M][M][M]"
3000 RETURN
3010 PRINT "[N][N][N][N][N]"
3020 PRINT "[V]         [B]"
3030 PRINT "[V][T]   [T][B]"
3040 PRINT "[V][T]   [T][B]"
3050 PRINT "[V][T]   [T][B]"
3060 PRINT "[V]         [B]"
3070 PRINT "[M][M][M][M][M]"
3080 RETURN
```



Alarm Clock



Deep within your computer is an extremely accurate clock. The Atari operating system goes upon a round of duties 3600 times a second, refreshing each dot on the screen, checking for input, carrying out your orders, etc. You can get a bench mark speed from any computer by putting in a delay loop like:

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

Your Atari will crunch this in about 11 seconds. Add a few more 0's and your warranty will expire before the computer is done. Put a loop like this on a mainframe and before your finger is off the key the cursor will say "OK." The Atari has plenty of time to draw a numeral in large graphic characters once every second. The clock program PEEKs into a monitor routine in the Atari that is called the frame counter. This counter is timed to 1/60th of a second to match the number of frames of a standard monitor or TV screen. Lines 1660 to 1670 figure out a high byte, middle byte, and low byte address for minutes, seconds, and hours. These bytes are POKEd into memory locations 20,19, and 18. The computer automatically updates these locations. Even if you stop and reload the clock program, the time will stay correct until you set new values. The time is read in line 1870 where the time locations are PEEKed at. The numerals are created by plotting a box-like figure 8 and blacking out appropriate segments of the box. Digital watch numerals work this way as well. The balance of the program remembers the number of days in a month and keeps you from entering a date like December 40. The months are written out when you input a number, and you can only enter Feb. 29 on a real leap year.

```
900 REM
910 REM ***************************************
920 REM *                                     *
930 REM *           ALARM CLOCK               *
940 REM *                                     *
950 REM ***************************************
960 REM
1000 DIM S(7),N(12),A$(9)
1010 N1=1
1020 N2=1
1030 N3=1980
1040 Q=0:GR.0
1050 PRINT "}"
1060 PRINT
1070 PRINT
1080 PRINT " TIMER"
1090 PRINT
1100 PRINT " DIGITAL CLOCK/CALENDAR"
1110 PRINT
1120 PRINT
1130 PRINT
1140 PRINT " OPTIONS"
1150 PRINT
1160 PRINT
1170 PRINT " 1 - SET DATE"
1180 PRINT " 2 - SET TIME"
1190 PRINT " 3 - SET TICK"
1200 PRINT " 4 - SET ALARM"
1210 PRINT " 5 - DISPLAY TIME"
1220 PRINT
1230 PRINT
1240 PRINT
1250 PRINT " WHICH OPTION"
1260 INPUT O
1270 PRINT "}"
1280 IF O<1 OR O>5 THEN 1220
1290 ON O GOTO 1350,1580,1730,1300,1770
1300 PRINT "ALARM TIME (H, M, S)"
1310 INPUT H1,M1,Z1
1320 IF H1>-1 AND H1<24 AND M1>-1 AND M1<60 AND Z1>-1 AND
     Z1<60 THEN 1050
1330 PRINT ILLEGAL TIME (24 HOUR FORMAT)!"
1340 GOTO 1300
1350 PRINT "CURRENT DATE (M, D, Y)"
```

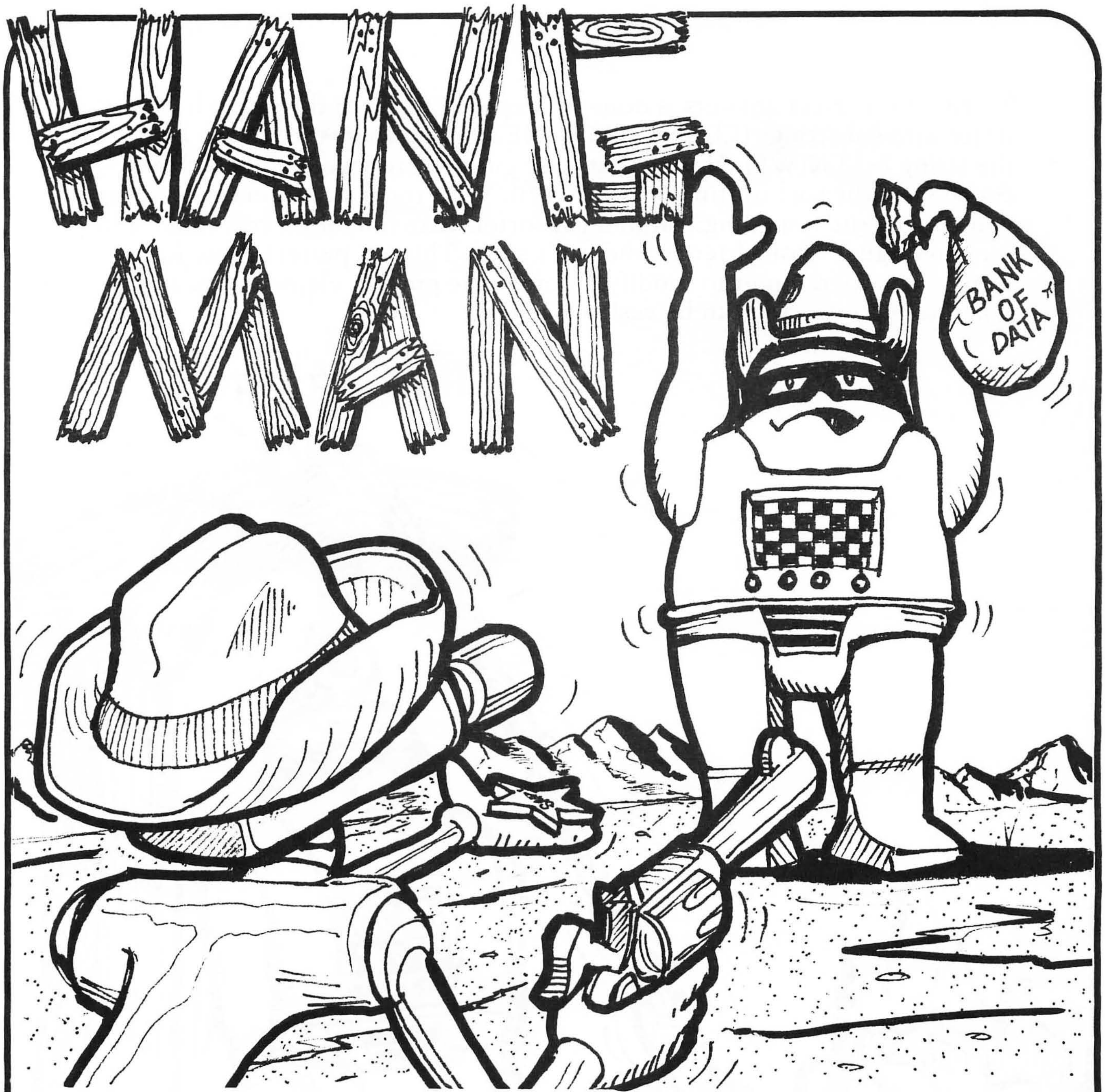
```
1360 INPUT N1,N2,N3
1370 IF N3<100 THEN N3=1900+N3
1380 N(1)=31
1390 N(2)=29
1400 N(3)=31
1410 N(4)=30
1420 N(5)=31
1430 N(6)=30
1440 N(7)=31
1450 N(8)=31
1460 N(9)=30
1470 N(10)=31
1480 N(11)=30
1490 N(12)=31
1500 IF N2>0 AND N2<N(N1)+1 THEN 1540
1510 GOSUB 2910
1520 PRINT A$;" ONLY HAS A MAX OF";N(N1);" DAYS!"
1530 GOTO 1350
1540 IF N1< >2 OR N2< >29 THEN 1050
1550 IF N3=INT(N3/4)*4 THEN 1050
1560 PRINT "THE NEXT LEAP YEAR IS";4+INT(N3/4)*4;"!"
1570 GOTO 1350
1580 PRINT "CURRENT TIME (H, M, S)"
1590 INPUT H,M,Z
1600 IF H>-1 AND H<24 AND M>-1 AND M<60 AND Z>-1 AND Z<60
     THEN 1630
1610 PRINT "ILLEGAL TIME (24 HOUR FORMAT)!"
1620 GOTO 1580
1630 SC=H*216000
1640 SC=SC+M*3600
1650 SC=SC+Z*60
1660 HB=INT(SC/65536)
1670 MB=INT((SC-65536*HB)/256)
1680 LB=SC-256*MB-HB*65536
1690 POKE 20,LB
1700 POKE 19,MB
1710 POKE 18,HB
1720 GOTO 1050
1730 PRINT "DO YOU WANT SOUND OF TICK (Y OR N)"
1740 INPUT A$
1750 IF A$="Y" THEN Q=1
1760 GOTO 1050
1770 GRAPHICS 4
1780 COLOR 1
```

```
1790 PLOT 27,18
1800 PLOT 27,22
1810 PLOT 43,18
1820 PLOT 48,22
1830 POKE 752,1
1840 H2=H+1
1850 M2=M+2
1860 UD=1
1870 SC=PEEK(18)*65536+PEEK(19)*256+PEEK(20)
1880 H=INT(SC/216000)
1890 IF H<25 THEN 1960
1900 UD=1
1910 GOSUB 3220
1920 POKE 20,0
1930 POKE 19,0
1940 POKE 18,0
1950 H=0:M=0:Z=0:GOTO 1990
1960 M=INT((SC-H*216000)/3600)
1970 Z=INT(SC-H*216000-M*3600)/60
1980 IF UD=0 THEN 2010
1990 GOSUB 3160
2000 UD=0
2010 IF H=H2 THEN 2050
2020 CX=11
2030 NUM=H
2040 GOSUB 2200
2050 IF M=M2 THEN 2090
2060 CX=32
2070 NUM=M
2080 GOSUB 2200
2090 CX=52
2100 NUM=Z
2110 GOSUB 2200
2120 IF H+M=0 AND H1>0 THEN GOSUB 3220
2130 IF Q=1 THEN SOUND 0,45,10,2
2140 IF H< >H1 OR M< >M1 OR Z-Z1<1 OR Z-Z1>10
     THEN SOUND 1,0,0,0:GOTO 2160
2150 SOUND 1,200,10,15
2160 SOUND 0,0,0,0
2170 H2=H
2180 M2=M
2190 GOTO 1870
2200 DIG=INT(NUM/10)
2210 GOSUB 2240
```

```
2220 CX=CX+6
2230 DIG=NUM-10*DIG
2240 REM DIGIT OUTPUT
2250 FOR K=1 TO 7
2260 S(K)=1
2270 NEXT K
2280 ON DIG+1 GOTO 2290,2310,2370,2400,2430,2470,2500,2530,2610,2580
2290 S(7)=0
2300 GOTO 2610
2310 S(1)=0
2320 S(4)=0
2330 S(5)=0
2340 S(6)=0
2350 S(7)=0
2360 GOTO 2610
2370 S(2)=0
2380 S(5)=0
2390 GOTO 2610
2400 S(5)=0
2410 S(6)=0
2420 GOTO 2610
2430 S(1)=0
2440 S(4)=0
2450 S(6)=0
2460 GOTO 2610
2470 S(3)=0
2480 S(6)=0
2490 GOTO 2610
2500 S(3)=0
2510 S(4)=0
2520 GOTO 2610
2530 S(1)=0
2540 S(5)=0
2550 S(6)=0
2560 S(7)=0
2570 GOTO 2610
2580 S(1)=0
2590 S(6)=0
2600 GOTO 2610
2610 COLOR 0
2620 FOR K=1 TO 7
2630 IF NOT S(K) THEN ON K GOSUB 2700,2730,2760,2790,2820,2850,2880
2640 NEXT K
```

```
2650 COLOR 1
2660 FOR K=1 TO 7
2670 IF S(K) THEN ON K GOSUB 2700,2730,2760,2790,2820,2850,2880
2680 NEXT K
2690 RETURN
2700 PLOT CX,30
2710 DRAWTO CX+4,30
2720 RETURN
2730 PLOT CX+4,20
2740 DRAWTO CX+4,30
2750 RETURN
2760 PLOT CX+4,10
2770 DRAWTO CX+4,20
2780 RETURN
2790 PLOT CX,10
2800 DRAWTO CX+4,10
2810 RETURN
2820 PLOT CX,10
2830 DRAWTO CX,20
2840 RETURN
2850 PLOT CX,20
2860 DRAWTO CX,30
2870 RETURN
2880 PLOT CX,20
2890 DRAWTO CX+4,20
2900 RETURN
2910 ON N1 GOTO 2920,2940,2960,2980,3000,3020,3040,
     3060,3080,3100,3120,3140
2920 A$="JANUARY"
2930 RETURN
2940 A$="FEBRUARY"
2950 RETURN
2960 A$="MARCH"
2970 RETURN
2980 A$="APRIL"
2990 RETURN
3000 A$="MAY"
3010 RETURN
3020 A$="JUNE"
3030 RETURN
3040 A$="JULY"
3050 RETURN
3060 A$="AUGUST"
3070 RETURN
```

```
3080 A$="SEPTEMBER"
3090 RETURN
3100 A$="OCTOBER"
3110 RETURN
3120 A$="NOVEMBER"
3130 RETURN
3140 A$="DECEMBER"
3150 RETURN
3160 GOSUB 2910
3170 PRINT "}"
3180 PRINT "            ";A$;" ";N2;", ";N3
3190 PRINT
3200 PRINT "ALARM SET FOR ";H1;":";M1;":";Z1
3210 RETURN
3220 N2=N2+1
3230 IF N2>N(N1) OR (N2=29 AND N3< >INT(N3/4)*4) THEN 3250
3240 GOTO 3160
3250 N1=N1+1
3260 N2=1
3270 IF N1<13 THEN 3160
3280 N1=1
3290 N3=N3+1
3300 GOTO 3160
3310 END
```

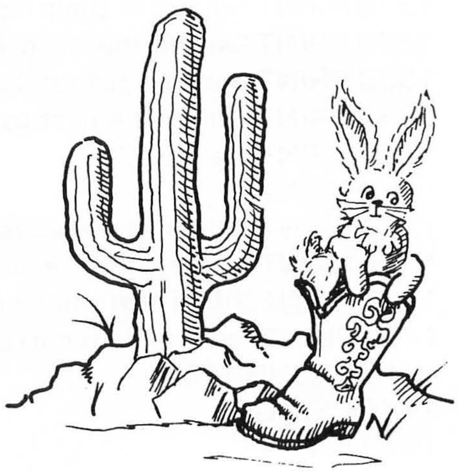


A classic word game that you can easily modify to contain different words or even a different graphic routine if you think hanging is too severe. One popular version of the game has a boat slowly sinking. If you miss too many letters, water goes over the gunwales and scuttles the boat. The ON GOTO command sends the program to the appropriate lines to draw the parts of the body and each section has a GOTO to send control back to the next guess. Letters are read off the keyboard directly, without the need to press RETURN after each guess. This is done by 'opening' the keyboard and using the GET command. There is a test to ignore the RETURN key if you do try to use it.

Testing for correct answers is done by stepping through the string letter by letter with substrings (I,I) with a FOR-NEXT loop that is set to the length of *the string* I=LEN(W$). Notice that your guesses are sorted alphabetically courtesy of the sort routine in 2080-2170. With the routines in this program you could write a spelling test or a list sorter. Late one night you could add a funeral dirge for your friends who can't spell. This is a perfect game for a beginning programmer to modify, because the graphic elements are all in separate sections, and can be easily changed.

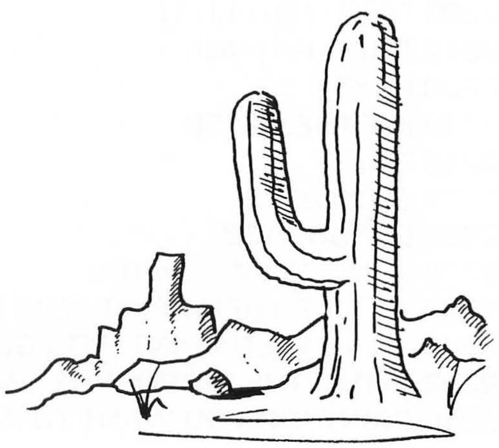
```
1ØØØ REM  ***********************************
1Ø1Ø REM *                                   *
1Ø2Ø REM *             HANG MAN              *
1Ø3Ø REM *                                   *
1Ø4Ø REM  ***********************************
1Ø5Ø REM
1Ø6Ø REM DIMENSION VARIABLES
1Ø7Ø DIM A$(2),G$(2Ø),T$(26),W$(2Ø)
1Ø8Ø REM SKIP SUBROUTINES
1Ø9Ø GOTO 12ØØ
11ØØ PRINT "}";
111Ø POSITION 12,2
112Ø PRINT "*** HANG MAN ***"
113Ø POSITION 2,5
114Ø RETURN
115Ø POKE 754,255
116Ø POSITION 2,23
117Ø PRINT "PRESS 'RETURN' TO CONTINUE";
118Ø IF PEEK(754)< >12 THEN 118Ø
119Ø RETURN
12ØØ GRAPHICS Ø
121Ø CLOSE #1
122Ø OPEN #1,4,Ø,"K:"
123Ø GOSUB 11ØØ
124Ø PRINT "In this game you are given the format"
125Ø PRINT "of a word. You try to spell out the"
126Ø PRINT "mystery word by guessing one letter"
127Ø PRINT "at a time."
128Ø PRINT
129Ø PRINT "If the letter is in the word then I"
13ØØ PRINT "will expose all occurrences of that"
131Ø PRINT "letter, in their correct position"
132Ø PRINT "within the word. If you guess a"
133Ø PRINT "letter not found in the word, then I"
134Ø PRINT "will add a part to the man in the"
135Ø PRINT "gallows."
136Ø PRINT
137Ø PRINT "When the man is complete, (head,"
138Ø PRINT "mouth, eyes, arms, and legs), he is"
139Ø PRINT "hung and you lose. To win, guess the"
14ØØ PRINT "entire word correctly."
141Ø GOSUB 115Ø
142Ø RESTORE
143Ø F=Ø
```

```
1440 C=0
1450 FOR I=1 TO INT(RND(1)*30)+1
1460 READ W$
1470 NEXT I
1480 G$=""
1490 FOR I=1 TO LEN(W$)
1500 G$(I,I)="-"
1510 NEXT I
1520 T$=""
1530 GRAPHICS 5
1540 SETCOLOR 0,3,10
1550 COLOR 1
1560 PLOT 3,0
1570 DRAWTO 79,0
1580 DRAWTO 79,39
1590 DRAWTO 3,39
1600 DRAWTO 3,0
1610 PLOT 38,7
1620 DRAWTO 38,4
1630 DRAWTO 40,4
1640 DRAWTO 40,3
1650 DRAWTO 8,3
1660 DRAWTO 8,37
1670 DRAWTO 9,37
1680 DRAWTO 9,31
1690 DRAWTO 67,31
1700 DRAWTO 67,37
1710 DRAWTO 68,37
1720 DRAWTO 68,30
1730 DRAWTO 9,30
1740 DRAWTO 9,4
1750 DRAWTO 38,4
1760 PLOT 10,11
1770 DRAWTO 16,5
1780 DRAWTO 17,5
1790 DRAWTO 10,12
1800 FOR I=32 TO 37
1810 COLOR 2-(I/2-INT(I/2))*2
1820 PLOT (I+1)-3*(I-32),I
1830 DRAWTO 2*I-21,I
1840 NEXT I
1850 IF W$=G$ THEN 2260
1860 PRINT "}Word :";G$
1870 PRINT "guesses:";T$
```

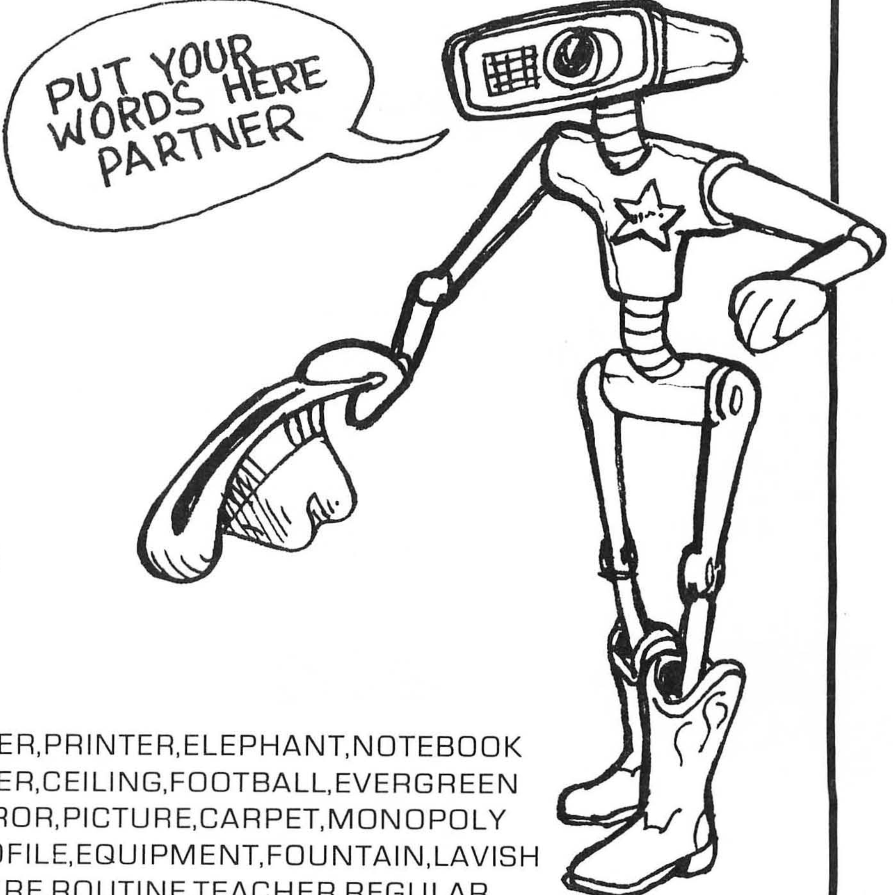


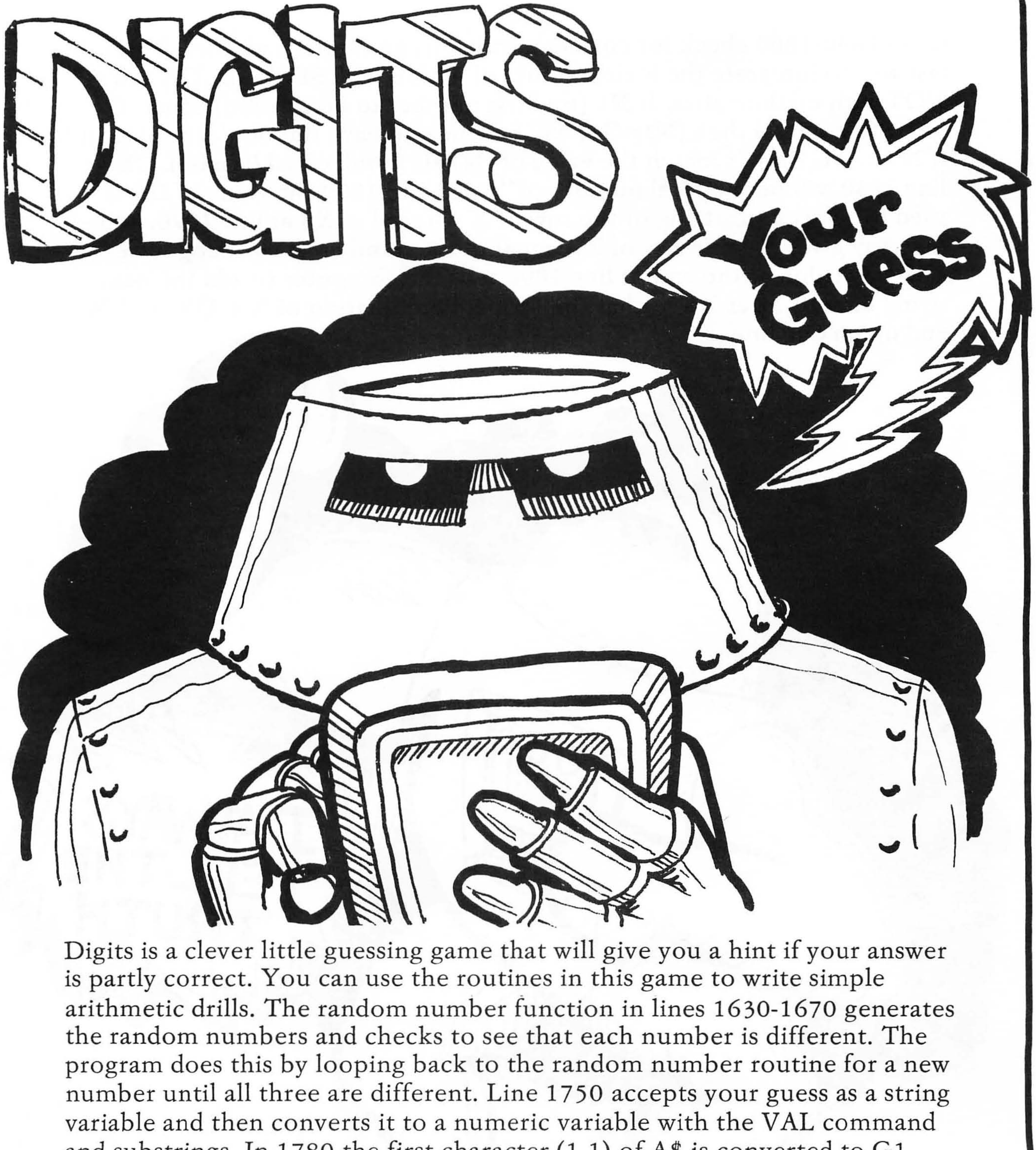
```
1880 IF F=1 THEN PRINT "THAT LETTER IS USED "
1890 IF F=3 THEN PRINT "YOU MISSED THAT ONE "
1900 IF F=2 THEN PRINT "YOU GOT ONE !!"
1910 IF F=0 THEN PRINT
1920 PRINT "Enter your guess ";
1930 GET #1,A
1940 IF A=155 THEN 1930
1950 A$=CHR$(A)
1960 F=0
1970 IF NOT LEN(T$) THEN 2010
1980 FOR I=1 TO LEN(T$)
1990 IF A$=T$(I,I) THEN F=1
2000 NEXT I
2010 IF F THEN 1850
2020 FOR I=1 TO LEN(G$)
2030 IF A$=G$(I,I) THEN F=1
2040 NEXT I
2050 IF F THEN 1850
2060 FOR I=1 TO LEN(W$)
2070 IF A$< >W$(I,I) THEN 2100
2080 F=2
2090 G$(I,I)=A$
2100 NEXT I
2110 IF F=2 THEN 1850
2120 L=LEN(T$)+1
2130 T$(L,L)=A$
2140 IF L=1 THEN 2250
2150 F=0
2160 FOR I=1 TO L-1
2170 IF T$(I,I)<T$(I+1,I+1) THEN 2220
2180 F=1
2190 A$=T$(I,I)
2200 T$(I,I)=T$(I+1,I+1)
2210 T$(I+1,I+1)=A$
2220 NEXT I
2230 IF F THEN 2150
2240 F=3
2250 GOTO 2370
2260 GOSUB 1100
2270 PRINT "GAME OVER."
2280 IF F< >2 THEN PRINT "YOU LOSE !!!"
2290 IF F=2 THEN PRINT "YOU GOT IT !!!"
2300 PRINT "THE WORD WAS ";W$
2310 PRINT "DO YOU WANT TO PLAY AGAIN Y/N ";
```

```
2320 GET #1,A
2330 A$=CHR$(A)
2340 IF A=155 THEN A$="Y"
2350 IF A$="Y" THEN 1420
2360 END
2370 C=C+1
2380 ON C GOTO 2390,2510,2560,2610,2660,2740,2810,2880,2980
2390 COLOR 2
2400 PLOT 37,8
2410 DRAWTO 39,8
2420 DRAWTO 41,10
2430 DRAWTO 41,12
2440 DRAWTO 39,14
2450 DRAWTO 37,14
2460 DRAWTO 35,12
2470 DRAWTO 35,10
2480 DRAWTO 37,8
2490 PLOT 38,15
2500 GOTO 1850
2510 COLOR 3
2520 PLOT 36,10
2530 PLOT 37,10
2540 PLOT 37,11
2550 GOTO 1850
2560 COLOR 3
2570 PLOT 40,10
2580 PLOT 39,10
2590 PLOT 39,11
2600 GOTO 1850
2610 COLOR 1
2620 PLOT 37,13
2630 PLOT 38,12
2640 PLOT 39,13
2650 GOTO 1850
2660 COLOR 1
2670 PLOT 37,16
2680 DRAWTO 37,22
2690 DRAWTO 39,22
2700 DRAWTO 39,16
2710 DRAWTO 38,16
2720 DRAWTO 38,22
2730 GOTO 1850
2740 COLOR 2
```



```
2750 PLOT 37,16
2760 DRAWTO 33,16
2770 DRAWTO 33,18
2780 DRAWTO 34,18
2790 DRAWTO 34,16
2800 GOTO 1850
2810 COLOR 2
2820 PLOT 39,16
2830 DRAWTO 43,16
2840 DRAWTO 43,18
2850 DRAWTO 42,18
2860 DRAWTO 42,16
2870 GOTO 1850
2880 COLOR 2
2890 PLOT 33,26
2900 DRAWTO 36,26
2910 DRAWTO 36,23
2920 DRAWTO 37,23
2930 DRAWTO 37,22
2940 DRAWTO 35,22
2950 DRAWTO 35,25
2960 DRAWTO 33,25
2970 GOTO 1850
2980 COLOR 2
2990 PLOT 43,26
3000 DRAWTO 40,26
3010 DRAWTO 40,23
3020 DRAWTO 39,23
3030 DRAWTO 39,22
3040 DRAWTO 41,22
3050 DRAWTO 41,25
3060 DRAWTO 43,25
3070 FOR I=0 TO 255 STEP 2
3080 SOUND 0,I,8,4
3090 NEXT I
3100 SOUND 0,0,0,0
3110 GOTO 2260
3120 DATA PENCIL,COMPUTER,PRINTER,ELEPHANT,NOTEBOOK
3130 DATA HANGMAN,POSTER,CEILING,FOOTBALL,EVERGREEN
3140 DATA YESTERDAY,MIRROR,PICTURE,CARPET,MONOPOLY
3150 DATA SCOUNDREL,PROFILE,EQUIPMENT,FOUNTAIN,LAVISH
3160 DATA RHYTHM,PLEASURE,ROUTINE,TEACHER,REGULAR
3170 DATA BARBECUE,BARRIER,PAVEMENT,THOUGHTFUL,MARRIAGE
```





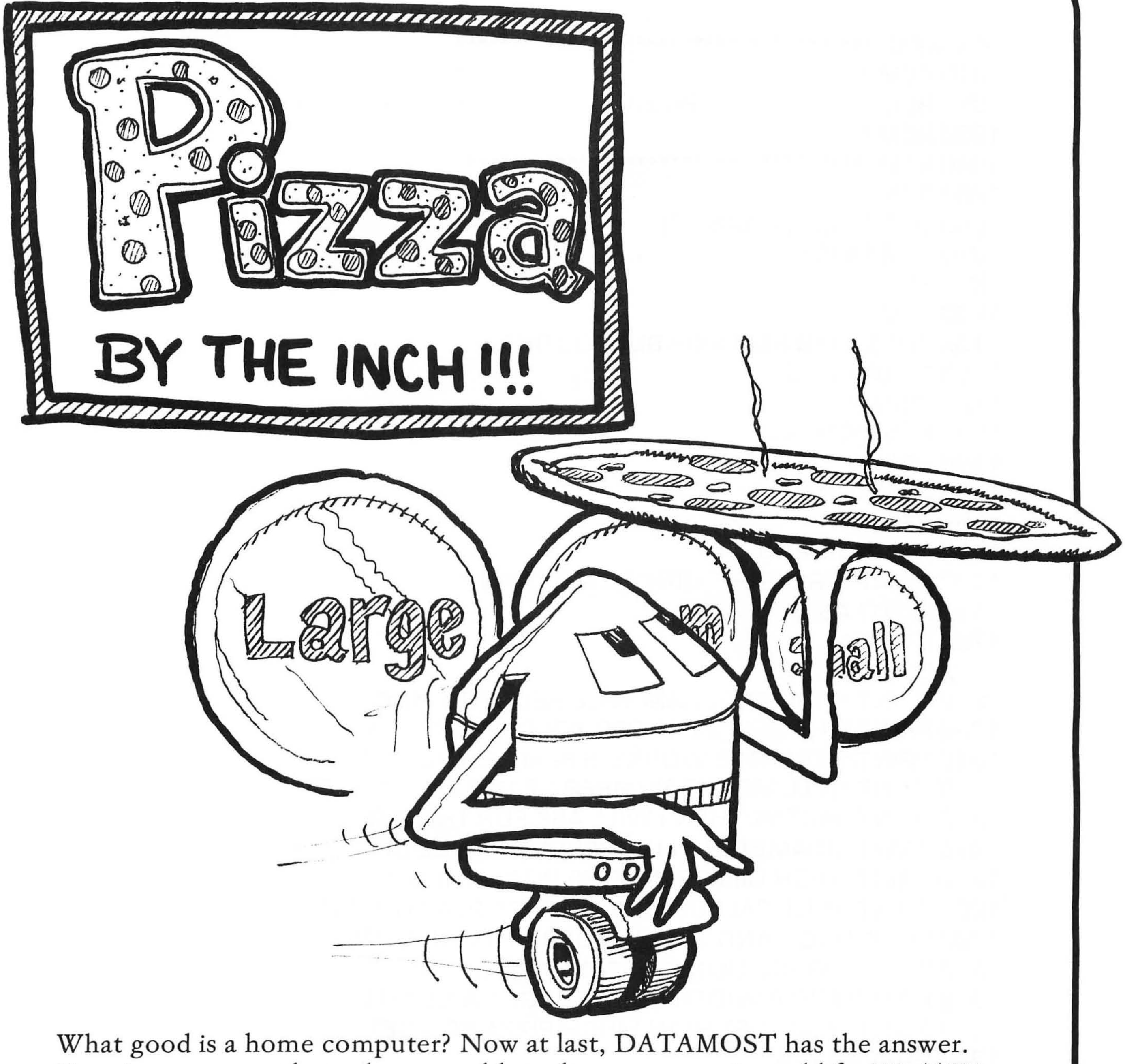
Digits is a clever little guessing game that will give you a hint if your answer is partly correct. You can use the routines in this game to write simple arithmetic drills. The random number function in lines 1630-1670 generates the random numbers and checks to see that each number is different. The program does this by looping back to the random number routine for a new number until all three are different. Line 1750 accepts your guess as a string variable and then converts it to a numeric variable with the VAL command and substrings. In 1780 the first character (1,1) of A$ is converted to G1, guess number ONE. A$(2,2) is converted to G2 etc. If you ask for a numeric (INPUT G1) directly and the player types "one" the program will crash. You should always enter numbers as strings, error check the input, and then use VAL to convert the sting to a numeric variable.

Lines 1830-1860 check for correct answers using Boolean algebra. This is a fast way to integrate the logical values of TRUE, FALSE, AND, OR and NOT with mathematics. If N1 (the first number to be guessed) equals G1 (your first guess) then (N1=G1) will be true and have the Boolean value of 1. If N1 and G1 don't match the value of (N1=G1) will equal 0 (false). CX in line 1830 will be the total number of "trues." In 1890 the value of CX is used to select a substring of the row of X's stored in X$ at line 1070, and that way gives the number of X's equal to the number of correct guesses. The semicolon at the end of line 1890 tells the computer to add the next string directly after X$ so that the proper combination of X's, O's, and -'s end up on one line.

```
1000 REM  ***********************************
1010 REM *                                  *
1020 REM *              DIGITS              *
1030 REM *                                  *
1040 REM  ***********************************
1050 REM DIMENSION VARIABLES
1060 DIM A$(10),X$(3),O$(3),L$(3)
1070 X$="XXX"
1080 O$="OOO"
1090 L$=" --- "
1100 GRAPHICS 0
1110 POSITION 14,3
1120 PRINT "*** DIGITS ***"
1130 POSITION 2,6
1140 PRINT "I will think of a number between 012"
1150 PRINT "and 987. Each digit in the number"
1160 PRINT "will be different from the other two"
1170 PRINT
1180 PRINT "The object of the game is to guess"
1190 PRINT "solution in as few tries as possible."
1200 POSITION 2,23
1210 PRINT "PRESS RETURN WHEN READY";
1220 INPUT A$
1230 PRINT "}";
1240 POSITION 14,3
1250 PRINT "*** DIGITS ***"
1260 POSITION 2,6
1270 PRINT "After each guess, I will print out a"
1280 PRINT "hint line as follows :"
1290 PRINT
1300 PRINT ">For each digit correct and in the"
1310 PRINT " correct position, I will print an 'x'"
1320 PRINT ">For each digit correct but not in"
1330 PRINT " the correct position, I will print"
1340 PRINT " an 'O' "
1350 PRINT
1360 PRINT ">For each totaly incorrect digit, I"
1370 PRINT " will print a '–' "
1380 PRINT
1390 PRINT " Play will continue untill you guess"
1400 PRINT " the correct number. To quit early,"
1410 PRINT " simply enter 'STOP' for your guess."
1420 POSITION 2,23
1430 PRINT "PRESS RETURN TO CONTINUE";
```

```
1440 INPUT A$
1450 PRINT "}";
1460 POSITION 2,4
1470 PRINT "Here is a table to help you"
1480 PRINT "understand the instructions."
1490 PRINT :PRINT
1500 PRINT "ANSWER              GUESS"
1510 PRINT "-------------------------"
1520 PRINT " 065                708 O--"
1530 PRINT " 562                463 X--"
1540 PRINT " 918                890 OO-"
1550 PRINT " 390                305 XO-"
1560 PRINT " 271                721 XOO"
1570 PRINT " 425                780 ---"
1580 PRINT
1590 POSITION 2,23
1600 PRINT "PRESS RETURN TO CONTINUE";
1610 INPUT A$
1620 NG=0
1630 N1=INT(RND(1)*10)
1640 N2=INT(RND(1)*10)
1650 IF N2=N1 THEN 1640
1660 N3=INT(RND(1)*10)
1670 IF N3=N1 OR N3=N2 THEN 1660
1680 PRINT "}";
1690 POSITION 14,4
1700 PRINT "*** DIGITS ***"
1710 PRINT :PRINT
1720 PRINT "Okay, I've got a number . . . "
1730 PRINT
1740 PRINT "What is your guess ";
1750 INPUT A$
1760 IF A$="STOP" THEN 1940
1770 IF LEN(A$)< >3 THEN PRINT "TYPE ONLY 3 DIGITS . . . }":GOTO 1740
1780 G1=VAL(A$(1,1))
1790 G2=VAL(A$(2,2))
1800 G3=VAL(A$(3,3))
1810 IF G1=G2 OR G2=G3 OR G1=G3 THEN PRINT
     "PLEASE TYPE THREE DIFFERENT DIGITS":GOTO 1740
1820 CX=0:CO=0:CL=0
1830 CX=(N1=G1)+(N2=G2)+(N3=G3)
1840 CO=(N1=G2)+(N1=G3)
1850 CO=CO+(N2=G1)+(N2=G3)
1860 CO=CO+(N3=G1)+(N3=G2)
```

```
1870 CL=3-(CX+CO)
1880 PRINT "For your guess of ";A$;" I hint ";
1890 IF CX>0 THEN PRINT X$(1,CX);
1900 IF CO>0 THEN PRINT O$(1,CO);
1910 IF CL>0 THEN PRINT L$(1,CL);
1920 NG=NG+1:PRINT
1930 IF CX< >3 THEN 1740
1940 PRINT "THE GAME IS OVER . . . ":PRINT
1950 IF CX=3 THEN PRINT "You guessed it in only ";NG;" trys !"
1960 IF CX<3 THEN PRINT "The correct answer was ";N1;N2;N3
1970 PRINT
1980 PRINT "WOULD YOU LIKE TO PLAY AGAIN Y/N ";
1990 INPUT A$
2000 IF LEN(A$)=0 THEN 2020
2010 IF A$(1,1)="Y" THEN 1620
2020 END
```

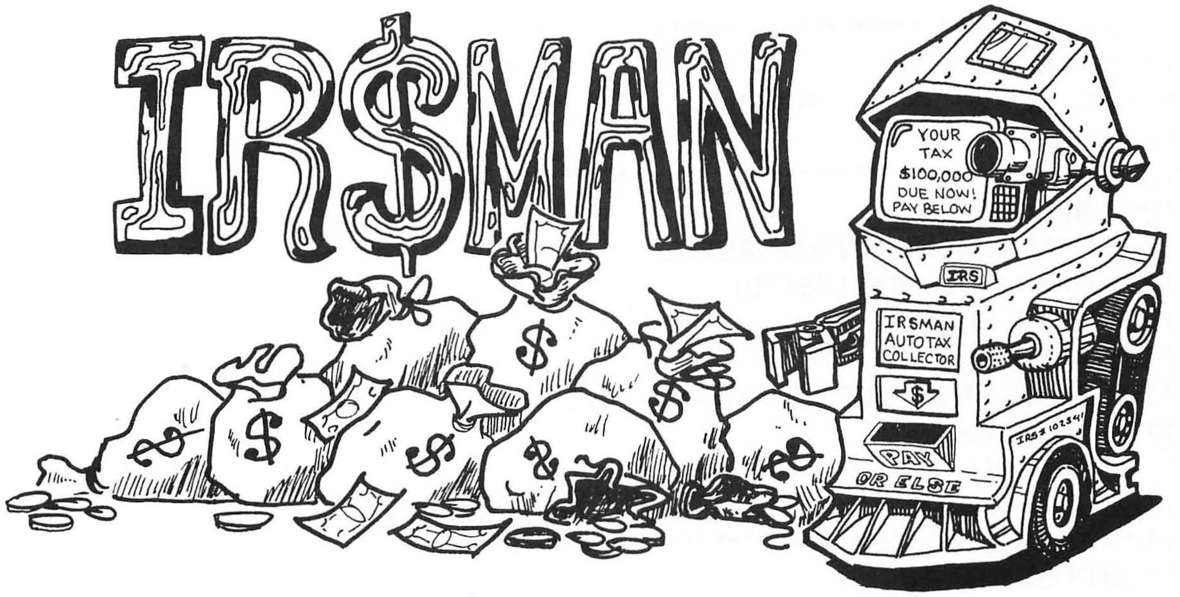


What good is a home computer? Now at last, DATAMOST has the answer. Here is a program that solves a problem that comes up in real life ALL THE TIME. Is a nine inch pizza for $4.95 worth as much as a 12 inch pizza for $6.50? The program is very sophisticated. We have considered that the outside crust occupies relatively more space on a small pizza and therefore displaces more of the goodies. The calculations will allow you to compare up to ten pizzas both ways! This program has screen formats, subroutines, FOR NEXT loops and scads of other vital programming tools. If every real life problem were attacked with powerful tools like this, you would get nothing done.

```
1ØØØ REM ************************************
1Ø1Ø REM *                                  *
1Ø2Ø REM *            PIZZA                 *
1Ø3Ø REM *                                  *
1Ø4Ø REM ************************************
1Ø5Ø REM
1Ø6Ø DIM D(1Ø),P(1Ø),A$(2Ø)
1Ø7Ø PI=3.14159
1Ø8Ø ST=1
1Ø9Ø REM
11ØØ GOTO 121Ø:REM SKIP SUBROUTINES
111Ø GRAPHICS Ø
112Ø PRINT "}"
113Ø POSITION 12,3
114Ø PRINT "*** PIZZA ***"
115Ø POSITION 2,6
116Ø RETURN
117Ø POSITION 2,23
118Ø PRINT "PRESS 'RETURN' TO CONTINUE";
119Ø INPUT A$
12ØØ RETURN
121Ø GOSUB 111Ø
122Ø PRINT " THIS PROGRAM WILL HELP YOU FIND"
123Ø PRINT "THE BEST DEAL FOR YOUR PIZZA"
124Ø PRINT "HOW THIS WORKS IS SIMPLE, YOU JUST"
125Ø PRINT "TELL ME THE NUMBER OF DIFFERENT SIZE"
126Ø PRINT "PIZZAS. THEN I WILL ASK FOR THE SIZE"
127Ø PRINT "(DIAMETER IN INCHES) AND PRICE OF"
128Ø PRINT "EACH DIFFERENT SIZE PIZZA. THEN I"
129Ø PRINT "WILL CALCULATE THE PRICE PER SQUARE"
13ØØ PRINT "INCH AND THE NUMBER OF INCHES OF"
131Ø PRINT "CRUST. USING A STANDARD OF ";ST
132Ø PRINT "FOR A WIDTH OF THE EDGE I WILL TELL"
133Ø PRINT "YOU JUST HOW MUCH PIZZA YOU GET."
134Ø GOSUB 117Ø
135Ø GOSUB 112Ø
136Ø PRINT "HOW MANY DIFFERENT SIZE PIZZAS";
137Ø INPUT A$
138Ø IF A$="" THEN A$="Ø"
139Ø NM=VAL(A$)
14ØØ IF NM=Ø THEN END
141Ø IF NM>1Ø THEN PRINT "1Ø PIZZAS MAX.":GOTO 136Ø
142Ø FOR X=1 TO NM
143Ø GOSUB 112Ø
```

```
1440 PRINT "FOR PIZZA NUMBER ";X
1450 PRINT
1460 PRINT "WHAT IS THE SIZE (DIAMETER IN INCHES)"
1470 PRINT "ENTER:";
1480 INPUT A$
1490 IF A$="" THEN A$="0"
1500 D(X)=VAL(A$)
1510 PRINT
1520 PRINT "WHAT IS THE PRICE"
1530 PRINT "ENTER:";
1540 INPUT A$
1550 IF A$="" THEN A$="0"
1560 P(X)=VAL(A$)
1570 PRINT
1580 NEXT X
1590 PRINT "}"
1600 PRINT
1610 PRINT " *** PIZZA ***"
1620 PRINT
1630 PRINT "N ....... NUMBER"
1640 PRINT "SZ ...... SIZE"
1650 PRINT "PR ...... PRICE"
1660 PRINT "PR/SQ' .. PRICE PER SQUARE INCH W/EDGE"
1670 PRINT "PR/SQO .. PRICE PER SQUARE INCH WO/EDGE"
1680 PRINT
1690 PRINT "N SZ PR PR/SQ' PR/SQO"
1700 PRINT
1710 FOR X=1 TO NM
1720 Y=10+X
1730 POSITION 2,Y
1740 PRINT X;
1750 POSITION 5,Y
1760 PRINT D(X);
1770 POSITION 9,Y
1780 PRINT P(X);
1790 POSITION 15,Y
1800 R=D(X)/2
1810 A=PI*R*R
1820 P=INT(100*P(X)/A+0.5)/100
1830 PRINT P;
1840 POSITION 23,Y
1850 R=R-ST
1860 A=PI*R*R
1870 P=INT(100*P(X)/A+0.5)/100
```

```
1880 PRINT P;
1890 NEXT X
1900 GOSUB 1170
1910 GOTO 1350
```



IRS Man is a fun game that will teach you about factors. Everyone wants to beat the IRS Man, and now here's your chance! To play, choose a number (we'll call it X), and the numbers 1 through X will appear. Each time you remove a number from the list, all of the factors of that number still on the list go to the IRS Man. The object of the game is to garner as much money as possible, while being as stingy as possible with the IRS Man . If you select twelve numbers to play with 1 2 3 4 5 6 7 8 9 10 11 12 will appear on the screen. If you play the number 12 first, the IRS man will get 6,2,4,3 and 1 (6x2=12, 3x4=12, 1x12=12) for a score of 16 to your 12. The board will then display 11 10 9 8 7 5. As you will note, the only remaining number on the list which has a factor displayed is 10 (the remaining factor is 5). Only a number that has a factor to give to the IRS man can be removed from the list, and all the numbers that cannot be factored go to him as well. After he has scooped these up the final score is 22 for you and 56 for the IRS. Talk about realism!

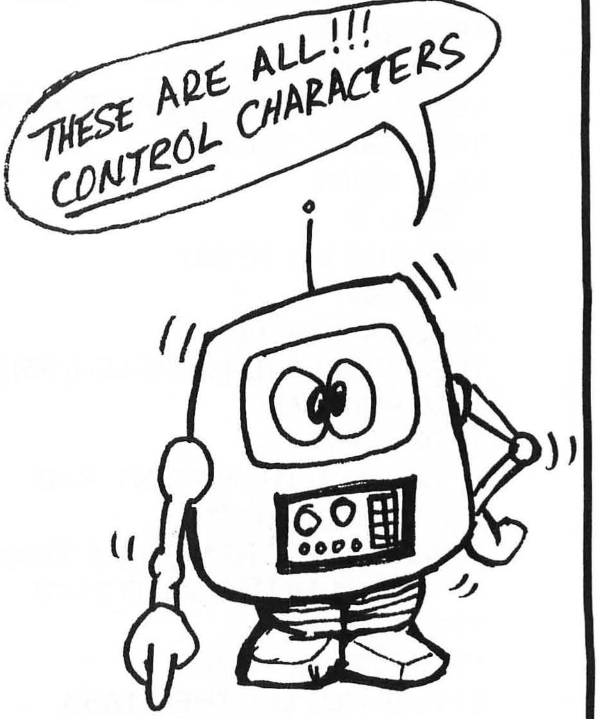
You can shelter some of your cash with strategy. Use the universal factor 1 to remove the largest prime number, 11. If you choose 8 next you won't give away the 6 or the 3 to the IRS. Choose 9 next and only the 3 goes to IRS. Now is the time to choose 12 because only the 6 is left to pay off with. This leaves 7,5 and 10 on the screen. Select 10; 5 and 7 go to our greedy advisary but we still win handily. 16 for the IRS and 40 for you. It probably means you'll be audited!

Think you've got it? Play with a bigger stake and both you and the computer will be pondering the moves for a long time.

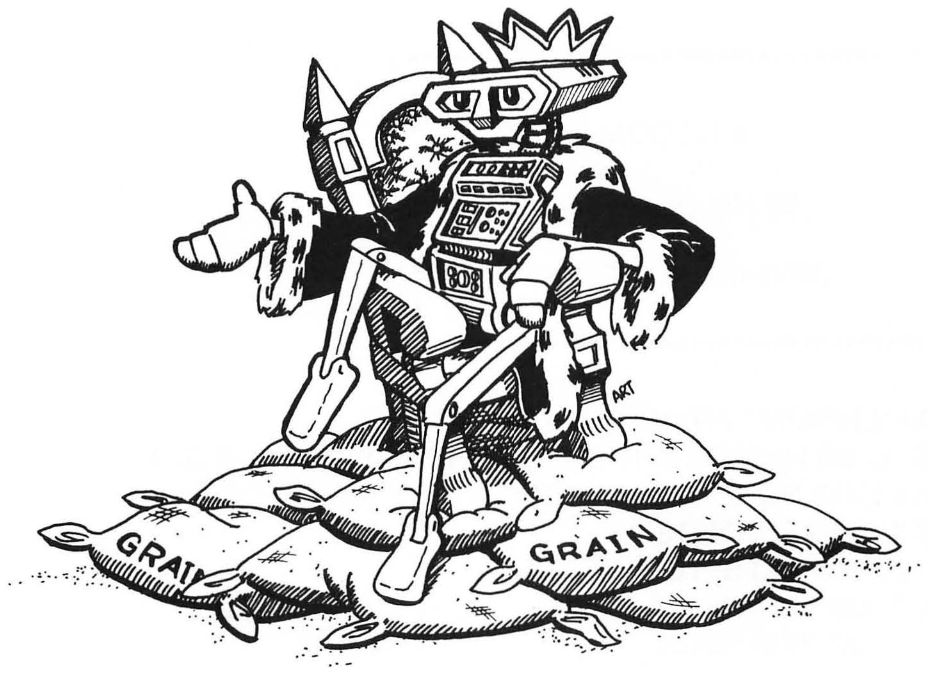
```
1000 REM  **********************************
1010 REM *                                 *
1020 REM *              IRS MAN            *
1030 REM *                                 *
1040 REM  **********************************
1050 REM
1060 REM DIMENSION VARIABLES
1070 DIM A$(20),LI(50),B$(20)
1080 GOTO 1130
1090 POSITION 2,23
1100 PRINT "Press 'RETURN' when ready";
1110 INPUT A$
1120 RETURN
1130 GRAPHICS 0
1140 POSITION 14,3:PRINT "*** IRS Man ***"
1150 POSITION 2,6
1160 PRINT "This is the game of IRS Man.
1160 To win,"
1170 PRINT "you try to accumulate more money than"
1180 PRINT "your nemesis, the IRS Man."
1190 PRINT
1200 PRINT "Give me a number between 1 and 50."
1210 PRINT "I will display a consecutive number"
1220 PRINT "string starting at 1, and continuing"
1230 PRINT "through to the number you selected."
1240 PRINT "You will then choose how much money"
1250 PRINT "(which number) you want to remove"
1260 PRINT "from the list."
1270 GOSUB 1090
1280 PRINT " esc clear ";
1290 POSITION 14,3:PRINT "*** IRS MAN ***"
1300 PRINT
1310 PRINT "But, and here's the fun part, the"
1320 PRINT "IRS Man gets all of the remaining"
1330 PRINT "numbers on the list that are factors"
1340 PRINT "of the number you chose. That is how"
1350 PRINT "the IRS Man gets his money. If you"
1360 PRINT "choose 6, for example, the IRS Man "
1370 PRINT "gets (potentially 1,2, and 3)."
1380 PRINT
1390 PRINT "You cannot choose a number that has"
1400 PRINT "no remaining factors in the list,"
1410 PRINT "because you must always pay the IRS."
1420 PRINT
```

```
1430 PRINT "When you can no longer remove any of"
1440 PRINT "the remaining numbers from the list,"
1450 PRINT "the IRS Man claims all of the unused"
1460 PRINT "money (numbers) for himself."
1470 GOSUB 1090
1480 FOR I=1 TO 50
1490 SC=0:TA=0
1500 LI(I)=I
1510 NEXT I
1520 PRINT "esc clear";
1530 POSITION 14,3:PRINT "*** IRS MAN ***"
1540 PRINT
1550 PRINT "HOW MANY NUMBERS ";
1560 INPUT A$
1570 A=0
1580 IF LEN(A$) THEN A=INT(VAL(A$))
1590 IF A<1 OR A>50 THEN PRINT "< < < USE A NUMBER BETWEEN 1
     AND 50 > > >":GOTO 1550
1600 NU=A
1610 PRINT "esc clear";
1620 POSITION 14,3:PRINT "*** IRS MAN ***"
1630 PRINT "The list:"
1640 PRINT
1650 K=0
1660 FOR I=1 TO NU
1670 A$="      "
1680 B$=STR$(I)
1690 IF LI(I) THEN A$(6-LEN(B$),6)=B$
1700 PRINT A$;
1710 K=K+1
1720 IF K=5 THEN PRINT :K=0
1730 NEXT I:PRINT
1740 IF NU=1 THEN PRINT "Ooops, you can't get anything . . . ";
     TA=1:LI(1)=0:GOTO 2140
1750 F=0
1760 FOR I=2 TO NU
1770 IF NOT LI(I) THEN 1830
1780 FOR J=1 TO I-1
1790 IF NOT LI(J) THEN 1820
1800 IF LI(I)/J< >INT(LI(I)/J) THEN 1820
1810 F=1
1820 NEXT J
1830 NEXT I
1840 IF NOT F THEN 2140
```

```
1850 PRINT
1860 PRINT "The score is: IRS Man: ";TA
1870 PRINT "           YOU : ";SC
1880 POSITION 2,20
1890 PRINT "Which do you want";
1900 INPUT A$
1910 A=INT(VAL(A$))
1920 IF A>NU OR A<0 THEN A=0
1930 IF A<1 OR NOT LI(A) THEN PRINT "That is not available";:GOTO 1880
1940 SD=0:IF A=1 THEN 2000
1950 FOR I=1 TO A
1960 IF NOT LI(I) THEN 1990
1970 IF I=A THEN 1990
1980 IF A/I=INT(A/I) THEN SD=SD+I
1990 NEXT I
2000 IF SD THEN 2050
2010 PRINT
2020 PRINT "You can't have it. That leaves"
2030 PRINT "nothing for the IRS Man"
2040 GOTO 1610
2050 LI(A)=0
2060 SC=SC+A
2070 TA=TA+SD
2080 FOR I=1 TO A
2090 IF NOT LI(I) THEN 2120
2100 IF I=A THEN 2120
2110 IF A/I=INT(A/I) THEN LI(I)=0
2120 NEXT I
2130 GOTO 1610
2140 REM GAME OVER
2150 PRINT :PRINT
2160 PRINT "*** THE GAME IS OVER ***"
2170 FOR I=1 TO NU
2180 IF LI(I) THEN TA=TA+LI(I)
2190 NEXT I
2200 PRINT
2210 PRINT "The IRS Man:";TA
2220 PRINT "           YOU:";SC
2230 PRINT "[N][N][N][N][N][N][N][N][N][N][N][N][N][N][N][N][N][N][N]"
     :REM UNDERLINE
2240 PRINT
2250 IF TA>SC THEN PRINT "The IRS Man is the winner !!!"
2260 IF TA<SC THEN PRINT "You have beaten the IRS Man !!!"
2270 IF TA=SC THEN PRINT "It's unbelievable but it's a tie !!!"
```



```
2280 PRINT " esc clear ";
2290 PRINT
2300 PRINT "Do you want to play again Y/N ";
2310 INPUT A$
2320 IF LEN(A$) THEN IF A$(1,1)="Y" THEN 1480
2330 END
```



Kingdom

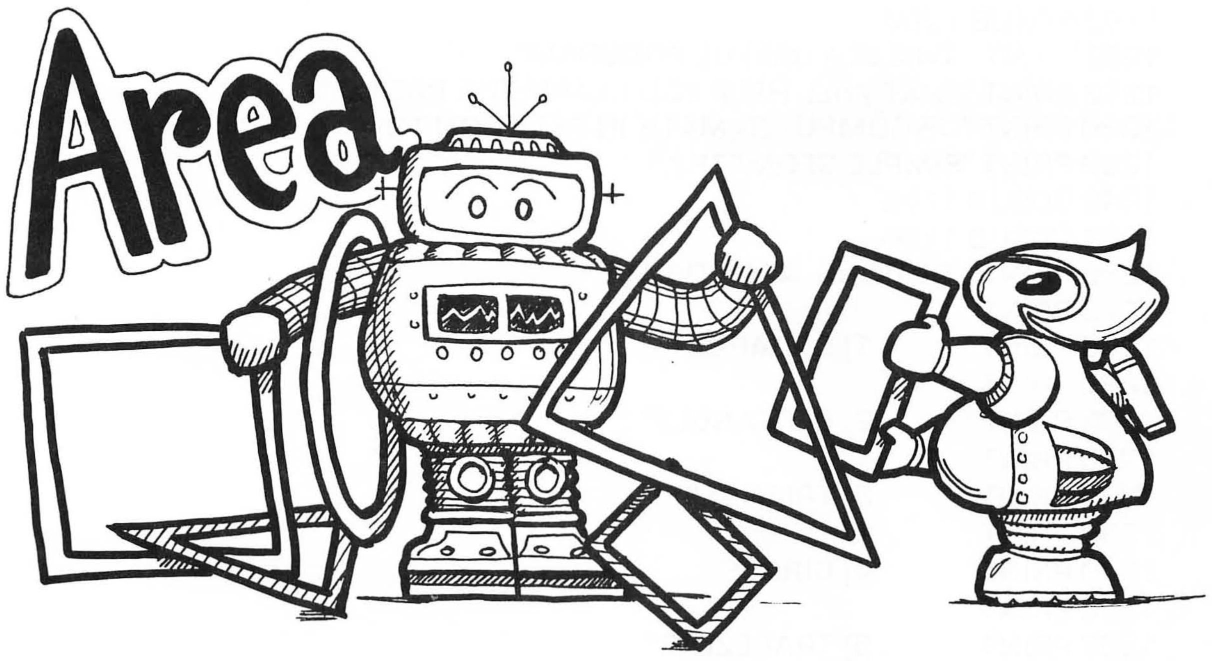
You are given a ten year reign as the king of Sumeria. You can buy land and hoard grain while your peasants starve, but a revolutionary tribunal will sit in judgement at the end of your reign. If the price of grain is high you should sell some land for grain, but watch out; rats are attracked to large amounts of grain and can literally eat up your profits. Lines 3630 to 3790 introduce random factors such as the price of land and the number of new inhabitants. The factors are intricately linked so that "nature" acts realistically. Occasionally an input will cause an unpredictable error, but you can resume the game with the CONTinue command. To understand how the variables work, change the beginning values in line 240, and change some of the random number statements. The game is an example of computer modeling, and you could use the framework to write any number of situations.

```
100 REM
110 REM ************************************
120 REM *                                  *
140 REM *            KINGDOM               *
150 REM *                                  *
160 REM *         By Hal Glicksman         *
170 REM *                                  *
180 REM *      COPYRIGHT DATAMOST          *
190 REM *                                  *
200 REM ************************************
210 REM
230 REM DIMENSION VARIABLES HERE
240 P=95:S=2800:H=3000:E=H-S:Y=3:A=H/Y:I=5:D=0:Z=0:Q=1
250 DIM ANS$(40),NU$(20),Q$(4),D$(4)
300 GOSUB 1000:REM INSTS
310 GOSUB 2000:REM SETUP
320 GOSUB 3000:REM PLAY!
330 GOSUB 4000:REM !END!
340 END
1000 REM ****
1001 REM *** INSTS
1002 REM ****
1010 PRINT "}":REM CLEAR SCREEN
1020 POSITION 12,3:PRINT "*** KINGDOM ***"
1030 PRINT :PRINT
1040 PRINT "This is a simulation of the kingdom of Sumeria."
1050 PRINT "The decisions which you make will affect the lives of hundreds
      of people"
1080 PRINT "You will be asked to make several key decisions each year, with
      each one being explained to you."
1100 POSITION 2,23:PRINT "Press 'RETURN' when ready.";
1130 INPUT ANS$
1990 RETURN
2000 REM ****
2001 REM *** SETUP
2002 REM ****
2010 REM *** SETUP
2020 REM ****
2040 REM YEAR SUBSTRINGS HERE
2050 RETURN
2100 NU$="first":SETCOLOR 2,3,1:SETCOLOR 4,3,3:RETURN
2110 NU$="second":SETCOLOR 2,1,1:RETURN
2120 NU$="third":SETCOLOR 2,4,1:RETURN
2130 NU$="fourth":SETCOLOR 2,0,1:RETURN
```

```
2140 NU$="fifth":SETCOLOR 2,7,1:RETURN
2150 NU$="sixth":SETCOLOR 2,0,1:RETURN
2160 NU$="seventh":SETCOLOR 2,3,1:RETURN
2170 NU$="eighth":SETCOLOR 2,4,1:RETURN
2180 NU$="ninth":SETCOLOR 2,1,1:RETURN
2190 NU$="tenth":SETCOLOR 2,3,2:RETURN
2200 NU$="final":SETCOLOR 2,4,2:RETURN
2210 GOTO 4100
3000 REM ****
3001 REM *** PLAY
3002 REM ****
3005 PRINT "}" PRINT "             *** KINGDOM ***"
3010 Z=Z+10
3020 GOSUB 2090+Z
3100 PRINT :PRINT
3120 PRINT "Hamurabi, I beg to report to you that in the ";NU$;" year of your reign, "
3125 PRINT D;" people starved."
3130 PRINT I;" new inhabitants came into the city."
3140 PRINT "Rats ate ";E;" bushels."
3280 P=P+I:IF Q=0 THEN P=INT(P/2):PRINT :PRINT "A horrible plague struck !!!
     Half of your people perished .... "
3285 PRINT
3290 PRINT "The population is ";P;"."
3295 PRINT "The city owns ";A;" acres."
3300 PRINT "You harvested ";Y;" bushels/acre."
3310 PRINT "You have ";S;" bushels in reserve."
3400 C=INT(RND(1)*10):Y=C+17
3410 PRINT "Land is trading at ";Y;" bushels/acre."
3420 PRINT "How many acres do you wish to buy?"
3430 INPUT Q$:IF Q$="" THEN Q$="0"
3431 Q=VAL(Q$)
3440 IF Q<0 OR Y*Q> THEN? "O wise Hamurabi please consider,":?
     "you only have";S;"bushels of grain.":GOTO 3420
3450 IF Q>0 THEN A=A+Q:S=S-Y*Q:C=0:GOTO 3500
3453 PRINT "Land is trading at ";Y;" bushels/acre."
3460 PRINT "How many acres of the royal domain":PRINT "should we sell";
3470 INPUT Q$:IF Q$="" THEN Q$="0"
3471 Q=VAL(Q$)
3480 IF Q>A THEN PRINT "O wisest of rulers, you only own ";A;" acres.":GOTO 3460
3490 A=A-Q:S=S+Y*Q:C=0
3500 PRINT "Of the ";S;" bushels remaining, how":PRINT "many
     do you wish to feed your":PRINT "people ";
3510 INPUT Q$:IF Q$="" THEN Q$="0"
3511 Q=VAL(Q$)
```

```
3520 IF (Q<1) THEN PRINT " O great one, please reconsider. The peasants
        will starve.":GOTO 3500
3530 IF Q>S THEN PRINT "Your beneficence exceeds the royal stores
        O mighty one.":GOTO 3500
3540 S=S-Q
3550 C=1
3560 PRINT "Of the ";A;" acres you now possess,"
3570 PRINT "how many do you wish to plant with grain";
3580 INPUT D$:IF D$="" THEN D$="0"
3581 D=VAL(D$)
3590 IF D<1 THEN PRINT "The people will starve O mighty one.":GOTO 3650
3600 IF (D/2)>S THEN PRINT "There is not seed enough for so
        much land.":GOTO 3560
3610 IF D>10*P THEN PRINT "You can only force one peasant to work
        ten acres of land."
3620 IF D>10*P THEN PRINT "Your population of ";P;"
        isn't big enought.":GOTO 3560
3630 S=S-INT(D/2)
3640 C=INT(RND(1)*5)+1
3650 Y=C
3660 H=D*Y
3670 E=0
3680 IF INT(C/2)*2=C THEN E=INT(S/C)
3690 S=S-E+H
3700 C=INT(RND(1)*5)+1
3710 I=INT(C*(20*A+S)/P/100+1)
3720 C=INT(Q/20)
3730 Q=INT(10*(2*RND(1)-0.3))
3740 IF P<C THEN D=0:GOTO 3010
3750 D=P-C
3760 IF D>0.5*P THEN 3800
3770 P1=((Z-1)*P1+D*100/P)/Z
3780 P=C
3790 D1=D1+D:GOTO 3010
3800 PRINT "O mighty one you have failed your people and have been deposed."
3810 PRINT "You have starved ";P;" peasants."
3820 WL=1
3830 FOR T=0 TO 255 STEP 4
3840 SOUND 0,T,10,10
3850 FOR PAUSE=1 TO 10:NEXT PAUSE
3860 NEXT T
3870 SOUND 0,0,0,0
3900 PRINT :PRINT
3910 GOTO 4300
```

```
3990 RETURN
4000 REM ***
4010 REM *** END ROUTINE
4020 REM ***
4100 PRINT "} *** This is the Report of ***          the Revolutionary Tribunal "
4105 PRINT :PRINT
4110 PRINT "In ten years of your rule,";P1*10;"% of the populous died each year."
4120 PRINT "A total of ";D1;" peasants perished."
4130 L=A/P
4140 PRINT "When you started, each peasant had ten acres."
4150 PRINT "Now each one has ";L;" acres."
4200 PRINT :PRINT
4220 IF P1>33 OR L<7 THEN PRINT "You have been deposed and beheaded,
       as befits a despot.":GOTO 4300
4230 IF P1>10 OR L<9 THEN PRINT "You are banished from the
       kingdom.":GOTO 4300
4240 IF P1>3 OR L<10 THEN PRINT "Only the upper classes wish
       you well.":GOTO 4300
4250 PRINT "You are wonderfully wise O great Hamurabi. Your reign will
       continue for many years"
4300 PRINT "Would you like to rule again";
4310 INPUT ANS$
4320 IF ANS$="" THEN RUN
4330 IF ANS$(1,1)="Y" THEN RUN
4340 END
4990 RETURN
```



If your family has been on your case for playing computer games you can write this educational program that will do a few plane geometry calculations. The program demonstrates how to format screens and input prompts. You can do all sorts of computations and put the results in a PRINT statement. The examples here are very easy, but you could put in very complex calculations without the program being any more complex.

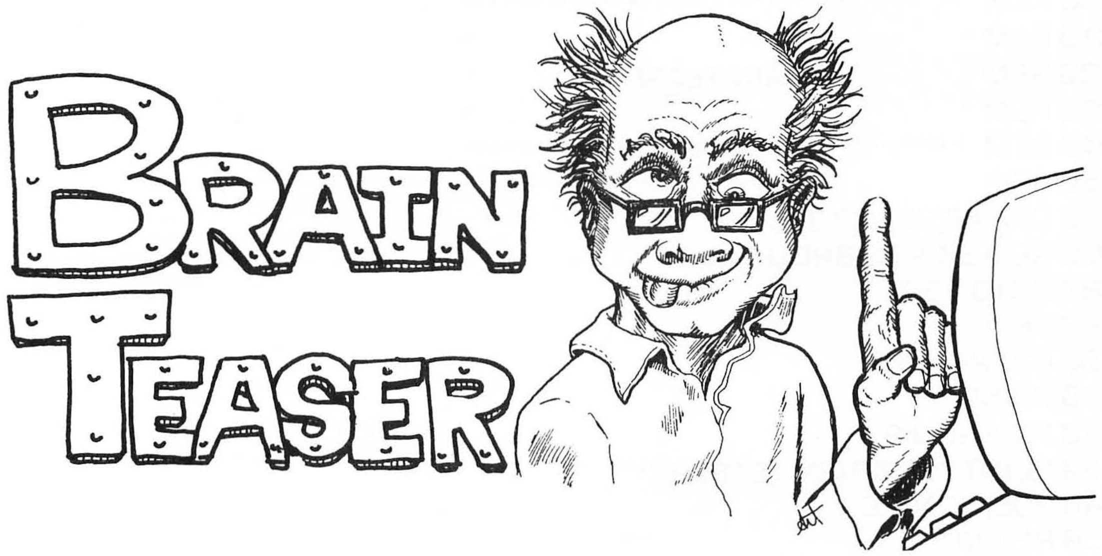
```
1000 REM  ***********************************
1010 REM *                                  *
1020 REM *              AREA                *
1030 REM *                                  *
1040 REM  ***********************************
1050 REM
1060 DIM A$(20)
1070 PI=3.14159
1080 GOTO 1190:REM SKIP SUBROUTINES
1090 GRAPHICS 0
1100 PRINT "}"
1110 POSITION 12,4
1120 PRINT "*** AREA ***"
1130 POSITION 2,8
1140 RETURN
1150 POSITION 2,23
1160 PRINT "PRESS 'RETURN' TO CONTINUE";
1170 INPUT A$:IF A$="" THEN A$="0"
1180 RETURN
```

```
1190 GOSUB 1090
1200 PRINT " THIS IS A USEFUL PROGRAM"
1210 PRINT "THAT WILL HELP YOU LEARN THE BASICS"
1220 PRINT "OF COMPUTER MATH IN RELATION TO"
1230 PRINT "SIMPLE GEOMETRY."
1240 GOSUB 1150
1250 GOSUB 1100
1260 PRINT " OPTIONS: AREA OF A"
1270 PRINT
1280 PRINT "        1] SQUARE"
1290 PRINT
1300 PRINT "        2] RECTANGLE"
1310 PRINT
1320 PRINT "        3] TRIANGLE"
1330 PRINT
1340 PRINT "        4] CIRCLE"
1350 PRINT
1360 PRINT "        5] TRAPEZOID"
1370 PRINT
1380 PRINT
1390 PRINT "        9] EXIT PROGRAM"
1400 PRINT
1410 PRINT "ENTER: ";
1420 GOSUB 1170
1430 A=INT(VAL(A$))
1440 IF A=9 THEN END
1450 IF A<1 OR A>5 THEN 1250
1460 ON A GOTO 1470,1670,1940,2210,2410
1470 GOSUB 1100
1480 PRINT "ENTER THE LENGTH OF A SIDE"
1490 PRINT "0 WILL RETURN TO MENU"
1500 PRINT
1510 PRINT "ENTER: ";
1520 GOSUB 1170
1530 S=VAL(A$):IF S<0 THEN 1470
1540 IF S=0 THEN 1250
1550 GOSUB 1100
1560 PRINT
1570 PRINT "A=S*S OR A=S*2"
1580 PRINT
1590 A=S*S
1600 PRINT "A SQUARE WITH DIMENSIONS OF"
1610 PRINT S;" UNITS BY ";S;" UNITS"
1620 PRINT "HAS AN AREA OF"
```

```
1630 PRINT A;" SQUARE UNITS."
1640 PRINT
1650 PRINT
1660 GOTO 1480
1670 GOSUB 1100
1680 PRINT "ENTER THE LENGTH OF SIDE 1"
1690 PRINT "0 WILL RETURN TO MENU"
1700 PRINT
1710 PRINT "ENTER: ";
1720 GOSUB 1170
1730 S1=VAL(A$):IF S1<0 THEN 1670
1740 IF S1=0 THEN 1250
1750 PRINT "ENTER THE LENGTH OF SIDE 2"
1760 PRINT "0 WILL RETURN TO MENU"
1770 PRINT
1780 PRINT "ENTER: ";
1790 GOSUB 1170
1800 S2=VAL(A$):IF S2<0 THEN 1670
1810 IF S2=0 THEN 1250
1820 GOSUB 1100
1830 PRINT
1840 PRINT "A=S1*S2"
1850 PRINT
1860 A=S1*S2
1870 PRINT "A RECTANGLE WITH DIMENSIONS OF"
1880 PRINT S1;" UNITS BY ";S2;" UNITS"
1890 PRINT "HAS AN AREA OF"
1900 PRINT A;" SQUARE UNITS."
1910 PRINT
1920 PRINT
1930 GOTO 1680
1940 GOSUB 1100
1950 PRINT "ENTER THE HEIGHT OF THE TRIANGLE"
1960 PRINT "0 WILL RETURN TO MENU"
1970 PRINT
1980 PRINT "ENTER: ";
1990 GOSUB 1170
2000 H=VAL(A$):IF H<0 THEN 1940
2010 IF H=0 THEN 1250
2020 PRINT "ENTER THE LENGTH OF THE BOTTOM"
2030 PRINT "0 WILL RETURN TO MENU"
2040 PRINT
2050 PRINT "ENTER: ";
2060 GOSUB 1170
```

```
2070 B=VAL(A$):IF B<0 THEN 1940
2080 IF B=0 THEN 1250
2090 GOSUB 1100
2100 PRINT
2110 PRINT "A=.5*B*H"
2120 PRINT
2130 A=0.5*B*H
2140 PRINT "A TRIANGLE WITH DIMENSIONS OF"
2150 PRINT H;" UNITS HIGH AND ";B;" UNITS WIDE"
2160 PRINT "HAS AN AREA OF"
2170 PRINT A;" SQUARE UNITS."
2180 PRINT
2190 PRINT
2200 GOTO 1950
2210 GOSUB 1100
2220 PRINT "ENTER THE RADIUS OF A CIRCLE"
2230 PRINT "0 WILL RETURN TO MENU"
2240 PRINT
2250 PRINT "ENTER: ";
2260 GOSUB 1170
2270 R=VAL(A$):IF R<0 THEN 2210
2280 IF R=0 THEN 1250
2290 GOSUB 1100
2300 PRINT
2310 PRINT "A=2*PI*R :REM PI=3.14159"
2320 PRINT
2330 A=2*PI*R
2340 PRINT "A CIRCLE WITH RADIUS OF"
2350 PRINT R;" UNITS"
2360 PRINT "HAS AN AREA OF"
2370 PRINT A;" SQUARE UNITS."
2380 PRINT
2390 PRINT
2400 GOTO 2220
2410 GOSUB 1100
2420 PRINT "ENTER THE LENGTH OF THE TOP"
2430 PRINT "0 WILL RETURN TO MENU"
2440 PRINT
2450 PRINT "ENTER: ";
2460 GOSUB 1170
2470 L1=VAL(A$):IF L1<0 THEN 2410
2480 IF L1=0 THEN 1250
2490 PRINT
2500 PRINT "ENTER THE LENGTH OF THE BOTTOM"
```

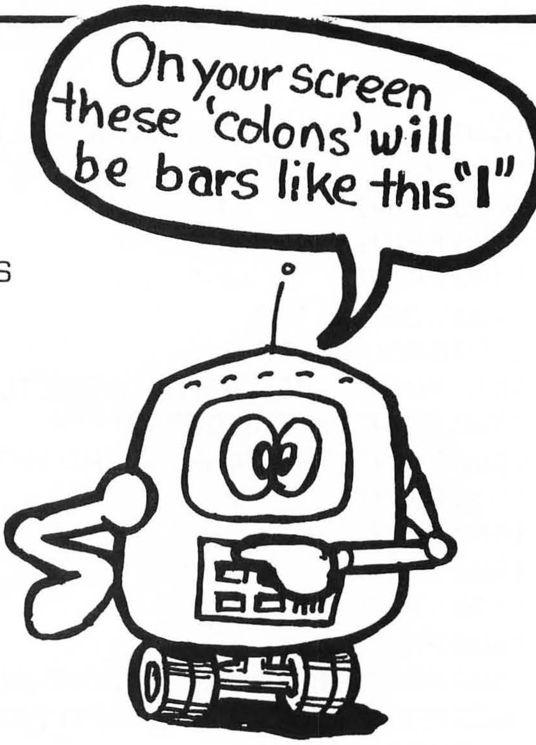
```
2510 PRINT "0 WILL RETURN TO MENU"
2520 PRINT
2530 PRINT "ENTER: ";
2540 GOSUB 1170
2550 L2=VAL(A$):IF L2<0 THEN 2410
2560 IF L2=0 THEN 1250
2570 PRINT
2580 PRINT "ENTER THE HEIGHT"
2590 PRINT "0 WILL RETURN TO MENU"
2600 PRINT
2610 PRINT "ENTER: ";
2620 GOSUB 1170
2630 H=VAL(A$):IF H<0 THEN 2410
2640 IF H=0 THEN 1250
2650 GOSUB 1100
2660 PRINT
2670 PRINT "A=L1*H+(L2-L1)/2*H"
2680 PRINT
2690 A=L1*H+(L2-L1)/2*H
2700 PRINT "A TRAPEZOID WITH DIMENSIONS OF"
2710 PRINT L1;" UNITS WIDE ON TOP"
2720 PRINT L2;" UNITS WIDE ON BOTTOM"
2730 PRINT H;" UNITS TALL"
2740 PRINT "HAS AN AREA OF"
2750 PRINT A;" SQUARE UNITS."
2760 PRINT
2770 PRINT
2780 GOTO 2420
```



A very simple premise results in a very complicated game. In Brain Teaser, simply select one of the squares that has a "+" (plus) sign in it. The square you chose will change to a minus sign and the other squares adjacent to your choice will flip to their opposite value. For some reason it seems easier to get all "–" (minus) squares and lose the game than it does to win. The print statements from 1920 to 2010 draw the grid of boxes that the plus and minus signs go into. If you type in the control characters indicated in the listings the grid will appear on your screen. The values for the plus and minus signs are stored in a two dimensional array, A(X,Y). Lines 2030 to 2090 have an inside and outside loop that counts the rows and columns of the grid. As the X's and Y's are counted off, the position statement places each sign in its proper box. The same X and Y also give the number of the array which holds the value for the sign that is drawn in the box. In line 2070 a flag is set to the number of plus signs. The flag is tested in lines 2100 and 2100. No pluses will send you to a "You Lose" line and nine pluses sends you to the winner's circle. Lines 2130 to 2200 are the player's move. Lines 2210 to 2410 juggles around the values in the arrays to set the new pluses and minuses. The ABS function converts all the negative values to positive. The values are flipped around like bowling pins in a jugglers act. You can pick the program apart with print statements between the lines to print out the values of the variables at each step.

```
1000 REM  ***********************************
1010 REM *                                  *
1020 REM *           BRAIN TEASER           *
1030 REM *                                  *
1040 REM  ***********************************
1050 REM
1060 DIM A$(20),A(3,3)
1070 REM SKIP SUBROUTINES
1080 GOTO 1200
1090 REM
1100 GRAPHICS 0
1110 PRINT "}";
1120 POSITION 9,4
1130 PRINT "*** BRAIN TEASER ***"
1140 POSITION 2,6
1150 RETURN
1160 POSITION 2,23
1170 PRINT "PRESS 'RETURN' TO CONTINUE";
1180 INPUT A$
1190 RETURN
1200 GOSUB 1100
1210 PRINT "I WILL DISPLAY A 3 BY 3 GAME BOARD"
1220 PRINT "WITH A + IN THE CENTER. IT WILL"
1230 PRINT "LOOK SOMTHING LIKE THIS."
1240 PRINT "           ---"
1250 PRINT "           -+-"
1260 PRINT "           ---"
1270 PRINT "THE TRICK IS TO MOVE SO THAT THE"
1280 PRINT "GAME BOARD LOOKS LIKE"
1290 PRINT
1300 PRINT " THIS +++ NOT LIKE THIS ---"
1310 PRINT "      +-+               ---"
1320 PRINT "      +++               ---"
1330 GOSUB 1160
1340 GOSUB 1110
1350 PRINT "YOU MAY ONLY MOVE TO AN OCCUPIED"
1360 PRINT "SPACE (A SPACE WITH A + ON IT)."
1370 PRINT "THEN THE ADJACENT SQUARES WILL INVERT"
1380 PRINT "AS WILL THE SQUARE YOU MOVE TO."
1390 PRINT "FOR EXAMPLE TAKE THE STARTING"
1400 PRINT "POSITION. ---"
1410 PRINT "          -+-"
1420 PRINT "          ---"
1430 PRINT " SINCE YOUR ONLY MOVE WOULD BE THE"
```

```
1440 PRINT "CENTER SQUARE THE GAME BOARD WOULD"
1450 PRINT "NOW LOOK LIKE THIS"
1460 PRINT "          -+-"
1470 PRINT "          +-+"
1480 PRINT "          -+-"
1490 GOSUB 1160
1500 GOSUB 1110
1510 PRINT "THEN IF YOU WERE TO MOVE IN THE"
1520 PRINT "SECOND POSITION:"
1530 PRINT "THE GAME BOARD WOULD LOOK LIKE THIS"
1540 PRINT
1550 PRINT "          +-+"
1560 PRINT "          +-+"
1570 PRINT "          -+-"
1580 PRINT
1590 PRINT "THEN IF YOU MOVE IN POSITION ONE"
1600 PRINT "THE GAME BOARD WOULD THEN LOOK LIKE"
1610 PRINT "THIS"
1620 PRINT " -++"
1630 PRINT " -++"
1640 PRINT " -+-"
1650 GOSUB 1160
1660 GOSUB 1110
1670 PRINT "SHOWN HERE ARE THE 3 DIFFERENT"
1680 PRINT "POSSIBLE MOVES (M) AND THE SQUARES"
1690 PRINT "THAT ARE EFFECTED."
1700 PRINT
1710 PRINT " M*- *M* -*-"
1720 PRINT " **- --- *M*"
1730 PRINT " --- --- -*-"
1740 PRINT
1750 PRINT "REMEMBER THAT THE SQUARE YOU MOVE TO"
1760 PRINT "WILL ALSO FLIP."
1770 PRINT
1780 PRINT "TO MOVE ENTER THE NUMBER OF THE BOX"
1790 PRINT "YOU WISH TO MOVE TO."
1800 PRINT "THE NUMBERS ARE:"
1810 PRINT "          123"
1820 PRINT "          456"
1830 PRINT "          789"
1840 GOSUB 1160
1850 GOSUB 1110
1860 FOR X=1 TO 3
1870 FOR Y=1 TO 3
```



```
1880 A(X,Y)=Ø
1890 NEXT Y
1900 NEXT X
1910 A(2,2)=1
1920 PRINT:REM DRAW ROWS OF BOXES
1930 PRINT
1940 PRINT "        [R][W][R][W][R][E]"
1950 PRINT "        : : : :"
1960 PRINT "        [A][R][R][R][D]"
1970 PRINT "        : : : :"
1980 PRINT "        [A][R][R][R][D]"
1990 PRINT "        : : : :"
2000 PRINT "        [Z][R][X][R][X][R][C]"
2010 PRINT
2020 F=Ø
2030 FOR X=1 TO 3
2040 FOR Y=1 TO 3
2050 POSITION X*2+14,Y*2+7
2060 IF A(X,Y)=Ø THEN PRINT "[R]";
2070 IF A(X,Y)>Ø THEN PRINT "";:F=F+1
2080 NEXT Y
2090 NEXT X
2100 IF F=Ø THEN 245Ø
2110 IF F=9 THEN 253Ø
2120 PRINT
2130 POSITION 2,2Ø
2140 PRINT "ENTER YOUR MOVE ";
2150 INPUT A$
2155 PRINT
2160 PRINT "          ";
2170 IF A$="" THEN A$="Ø"
2180 IF ASC(A$)<49 THEN 247Ø
2181 IF ASC(A$)>57 THEN 247Ø
2190 A=VAL(A$)
2200 IF A<1 OR A>9 THEN 213Ø
2210 Y=INT((A-1)/3)+1
2220 X=A-(Y-1)*3
2230 IF A(X,Y)=Ø THEN 242Ø
2240 A(X,Y)=ABS(A(X,Y)-1)
2250 IF X< >2 THEN 229Ø
2260 A(1,Y)=ABS(A(1,Y)-1)
2270 A(3,Y)=ABS(A(3,Y)-1)
2280 IF Y< >2 THEN 2Ø2Ø
2290 IF Y< >2 THEN 233Ø
```

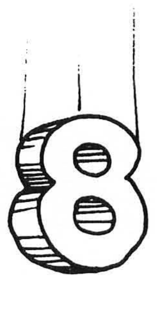
```
2300 A(X,1)=ABS(A(X,1)-1)
2310 A(X,3)=ABS(A(X,3)-1)
2320 GOTO 2020
2330 A(X,Y)=ABS(A(X,Y)-1)
2340 IF X=3 THEN X=2
2350 IF Y=3 THEN Y=2
2360 FOR X1=X TO X+1
2370 FOR Y1=Y TO Y+1
2380 A(X1,Y1)=ABS(A(X1,Y1)-1)
2390 NEXT Y1
2400 NEXT X1
2410 GOTO 2020
2420 POSITION 2,22
2430 PRINT "THAT SPACE IS EMPTY }";
2440 GOTO 2020
2450 POSITION 2,22
2460 PRINT "THE GAME IS OVER YOU LOSE"
2470 POSITION 2,23
2480 PRINT "WOULD YOU LIKE TO PLAY AGAIN Y/N ";
2490 INPUT A$
2500 IF A$="" THEN A$="N"
2510 IF A$(1,1)="Y" THEN 1850
2520 END
2530 POSITION 2,22
2540 PRINT "CONGRATULATIONS YOU WIN!"
2550 GOTO 2470
```

Numbers Away

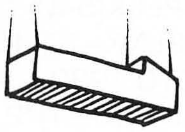
Numbers away is deceptively simple, but very hard to outwit. The object is to eliminate as many numbers as possible from the list before you get stymied. A pair of random numbers represents the roll of the of dice. The total on the dice must be removed from the list. There are multiple combinations of numbers that will match the dice on the first turn. If the first roll was 4+5 you could choose 1+8,2+7,3+6,4+5, or 9 by itself. Whatever number you choose is eliminated from the list and not available for the next guess. The trick is to save the numbers that are most likely to be useful in subsequent turns! Surviving even three rolls is an effort.

This program demonstrates the use of large characters (PRINT#6) and full screen graphics (GRAPHICS 2+16). The subroutine for generating the random number is at the beginning of the program. The flow of control goes around the subroutine by means of the GOTO statement in line 1080. This allows the subroutine to operate significantly faster than it would at the end of the program. Lines 1880-1920 show you how to read the function keys and use them in your programs. This program could be easily enhanced with sound routines and a flashier title screen. This is known in the trade as adding "bells and whistles."

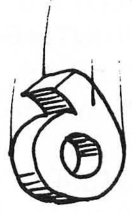
```
1000 REM  ***********************************
1010 REM *                                 *
1020 REM *          NUMBERS AWAY           *
1030 REM *                                 *
1040 REM  ***********************************
1050 REM
1060 REM DIMENSION VARIABLES
1070 DIM A$(20),L1(9),L2(9)
1080 GOTO 1210
1090 POSITION 2,23
1100 PRINT "Press return to continue";
1110 INPUT A$
1120 RETURN
1130 FOR J=1 TO INT(RND(1)*5)+8
1140 D1=INT(RND(1)*6)+1
1150 D2=INT(RND(1)*6)+1
1160 PLOT 7,3:PRINT #6;D1;
1170 PLOT 9,3:PRINT #6;D2;
1180 FOR DL=1 TO 10:NEXT DL
1190 NEXT J
1200 RETURN
1210 GRAPHICS 0
1220 POSITION 11,4
1230 PRINT "*** NUMBERS AWAY ***"
1240 POSITION 2,7
1250 PRINT "In this game, you will be presented"
1260 PRINT "with a list of numbers between"
1270 PRINT "1 and 9."
1280 PRINT
1290 PRINT "A pair of dice will be rolled and"
1300 PRINT "the total will be noted. You must"
1310 PRINT "remove, from the list, a combination"
1320 PRINT "of numbers whose total matches the"
1330 PRINT "total of the dice."
1340 PRINT
1350 PRINT "For example, if a seven was rolled,"
1360 PRINT "you could remove from the list"
1370 PRINT "(1,2,4), (1,6), (2,5), (3,4) or just"
1380 PRINT "plain (7)."
1390 GOSUB 1090
1400 PRINT " esc clear ";
1410 POSITION 11,4
1420 PRINT "*** NUMBERS AWAY ***"
1430 POSITION 2,7
```



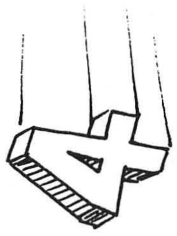
```
1440 PRINT "To move in the list, use the 'OPTION' "
1450 PRINT "key to move left the 'SELECT' key to"
1460 PRINT "move right. The number you are at"
1470 PRINT "will blink. To select a number"
1480 PRINT "press the 'START' key. When you"
1490 PRINT "select enough numbers to reach the"
1500 PRINT "total on the dice, I will roll the"
1510 PRINT "dice for you next try."
1520 PRINT
1530 PRINT "If your total goes over the the"
1540 PRINT "number, the list will be restored,"
1550 PRINT "and you will have to try again. To"
1560 PRINT "give up press the 'ESC' key."
1570 GOSUB 1090
1580 FOR I=1 TO 9
1590 L1(I)=I
1600 NEXT I
1610 GRAPHICS 2+16
1620 FOR I=1 TO 9
1630 PLOT (I-1)*2,1
1640 PRINT #6;I;
1650 NEXT I
1660 A=L1(1)
1670 FOR I=2 TO 9
1680 A=A+L1(I)
1690 NEXT I
1700 IF A>0 THEN 1720
1710 PLOT 0,8:PRINT #6;"YOU GOT THEM ALL !!":GOTO 2240
1720 GOSUB 1130
1730 ST=0
1740 PLOT 0,5
1750 PRINT #6;D1;"+";D2;"=";D1+D2;" ";
1760 PLOT 0,6
1770 PRINT #6;D1+D2;"-";ST;"=";D1+D2-ST;"  ";
1780 FOR I=1 TO 9
1790 L2(I)=0
1800 NEXT I
1810 TT=D1+D2
1820 NP=1
1830 BLNK=0
1840 IF L1(NP)>0 THEN 1870
1850 NP=NP+1:IF NP=10 THEN NP=1
1860 GOTO 1840
1870 IF PEEK(764)=28 THEN 2260
```



```
1880 KEY=PEEK(53279):IF KEY=7 THEN 1930
1890 IF KEY=6 THEN 2020
1900 IF KEY=3 THEN NP=NP-1:IF NP=0 THEN NP=9
1910 IF KEY=5 THEN NP=NP+1:IF NP=10 THEN NP=1
1920 IF L1(NP)=0 THEN 1900
1930 BLNK=ABS(BLNK-1)
1940 PLOT (NP-1)*2,1
1950 IF BLNK=0 THEN PRINT #6;" ";
1960 IF BLNK=1 THEN PRINT #6;NP;
1970 REM WAIT
1980 POKE 540,25
1990 IF PEEK(540) THEN 1990
2000 IF BLNK=1 THEN 1870
2010 GOTO 1930
2020 PLOT (NP-1)*2,1
2030 PRINT #6;" ";
2040 ST=ST+NP
2050 L1(NP)=0
2060 L2(NP)=1
2070 PLOT 0,6
2080 PRINT #6;TT;"-";ST;"=";TT-ST;" ";
2090 POKE 540,30
2100 IF PEEK(540) THEN 2100
2110 IF ST<TT THEN 1830
2120 IF ST=TT THEN 1660
2130 FOR I=1 TO 9
2140 IF NOT L2(I) THEN 2190
2150 L1(I)=I
2160 L2(I)=0
2170 PLOT (I-1)*2,1
2180 PRINT #6;I;
2190 NEXT I
2200 ST=0
2210 PLOT 0,6
2220 PRINT #6;TT;"-";ST;"=";TT-ST;"  ";
2230 GOTO 1830
2240 POKE 540,255
2250 IF PEEK(540) THEN 2250
2260 GRAPHICS 0
2270 POSITION 11,4
2280 PRINT "*** NUMBERS AWAY ***"
2290 POSITION 2,7
2300 GT=0
2310 FOR I=1 TO 9
```



```
2320 IF NOT L1(I) THEN GT=GT+I
2330 NEXT I
2340 PRINT
2350 PRINT "You got ";GT;" out of 45 ";
2360 PRINT
2370 PRINT "That's ";
2380 ON INT(GT/5) GOTO 2390,2410,2430,2450,2470,2490,2510,2530,2550
2390 PRINT "THE ABSLOUTE WORST !!!"
2400 GOTO 2570
2410 PRINT "EXTREMELY POOR !!!"
2420 GOTO 2570
2430 PRINT "TERRIBLE !!!"
2440 GOTO 2570
2450 PRINT "VERY BAD !!!"
2460 GOTO 2570
2470 PRINT "JUST SO-SO !!!"
2480 GOTO 2570
2490 PRINT "FAIR . . . "
2500 GOTO 2570
2510 PRINT "PRETTY GOOD . . . "
2520 GOTO 2570
2530 PRINT "GREAT !!!"
2540 GOTO 2570
2550 PRINT "PERFECT !!!!!!!"
2560 GOTO 2570
2570 PRINT "Would you like to play again Y/N ";
2580 INPUT A$
2590 IF LEN(A$) THEN IF A$(1,1)=  Y" THEN 1580
2600 END
```



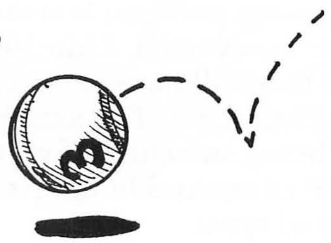
Test your reasoning ability on this mind expanding game. You are given a list of integers that must be unscrambled by reversing the order of the list. The boxes for the numbers are printed in lines 1800-1820, and remain on the screen when new numbers are printed in each loop. The POSITION command makes sure the numbers go into the proper places inside the boxes. The numbers that are scrambled and reversed are stored in a numeric array. The variable I in 1830-1860 determines the proper position for the number and the correct array element. Lines 1990-2040 do the reversing. The beginning position is stored in A. The numbers from A to the end of the line must be reversed. Line 1990 figures out the middle of the group of numbers that will be reversed. Lines 2000-2040 are a FOR-NEXT loop that swaps the numbers. The variable T is a temporary variable that stores the number being switched. If you want to see how this switching works, insert the STOP command in different parts of the program. When the program has stopped type:

```
FOR I=Ø TO 9 :PRINT L1(I):NEXTI and press RETURN
```

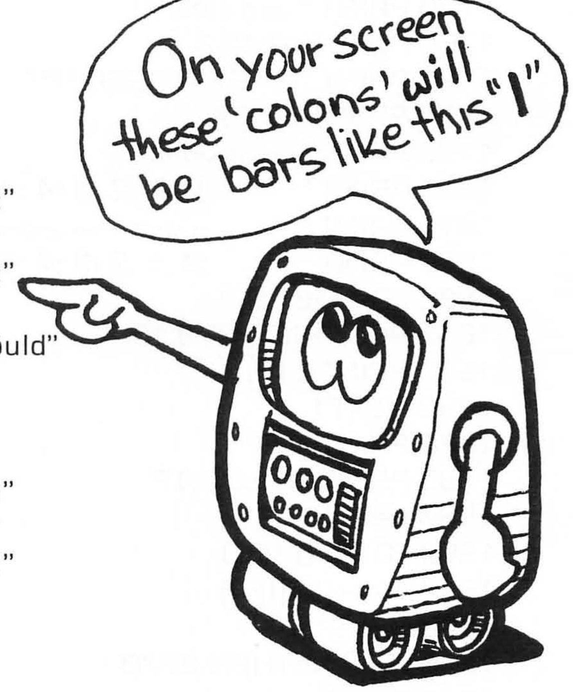
This is an immediate mode command, so everything has to be on one line to work. You can restart the game with the CONT command.

How does the program know when the numbers are in the correct order? This is done with a clever use of Boolian algebra in line 1910 A=A+(L1(I)=I). L1(I) is the array element and I is the position in the array. The numbers in the top row are the same as the positions. For every match between the array element and the position (L1(I)=I) will be true and equal to 1. A loop from 1900-1920 tests all 10 elements and adds up the number of "trues." If all 10 are true then 1930 sends you to the winner's circle.

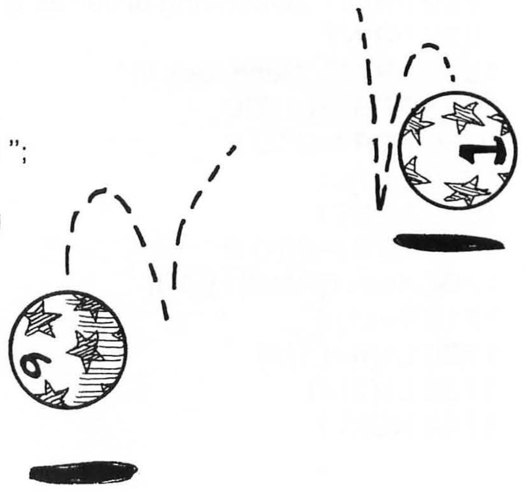
```
1ØØØ REM ***********************************
1Ø1Ø REM *                                 *
1Ø2Ø REM *           REVERSER              *
1Ø3Ø REM *                                 *
1Ø4Ø REM ***********************************
1Ø5Ø REM
1Ø6Ø REM DIMENSION VARIABLES
1Ø7Ø DIM A$(2Ø),L1(1Ø)
1Ø8Ø GOTO 113Ø
1Ø9Ø POSITION 2,23
11ØØ PRINT "Press return to continue ";
111Ø INPUT A$
112Ø RETURN
113Ø GRAPHICS Ø
114Ø POSITION 13,3
115Ø PRINT "*** REVERSER ***"
116Ø POSITION 2,6
117Ø PRINT "In this game, you are given a list of"
118Ø PRINT "numbers from Ø to 9. The list will"
119Ø PRINT "not be in sequence. It is your job"
12ØØ PRINT "to sort the numbers into ascending"
121Ø PRINT "order."
122Ø PRINT
123Ø PRINT "You arrange the list be reversing the"
124Ø PRINT "order of it. You input the starting"
125Ø PRINT "column that is to be reversed, and"
126Ø PRINT "that column, all the way through to"
127Ø PRINT "column nine, will be reversed."
128Ø GOSUB 1Ø9Ø
129Ø PRINT " esc clear ";
13ØØ POSITION 13,3
```



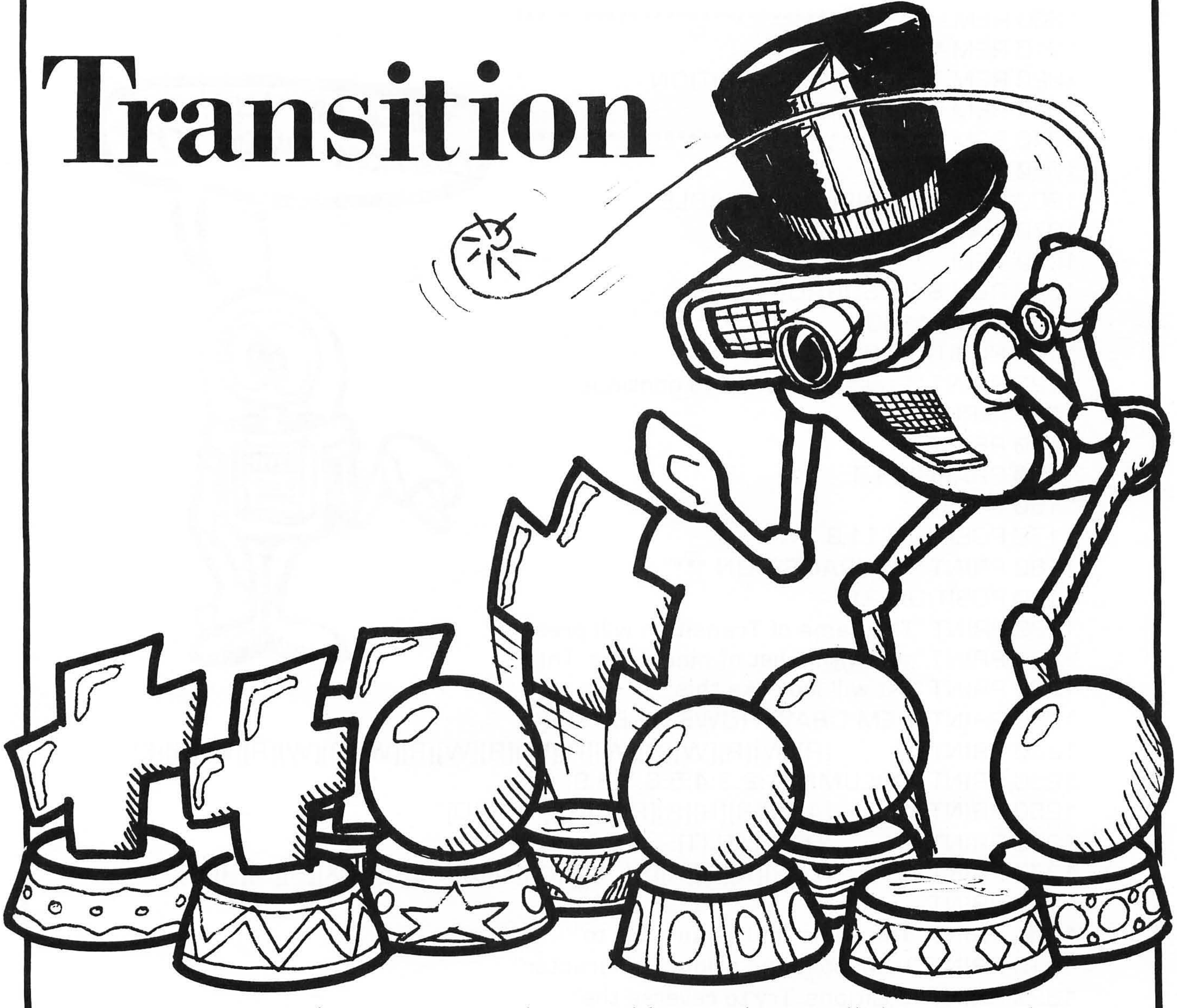
```
131Ø PRINT "*** REVERSER ***"
132Ø POSITION 2,7
133Ø PRINT "If you had this list:"
134Ø PRINT
135Ø PRINT "Positions:"
136Ø PRINT "          :Ø :1 :2 :3 :4 :5 :6 :7 :8 :9 :"
137Ø PRINT "          :--:--:--:--:--:--:--:--:--:--:"
138Ø PRINT "          :Ø :1 :9 :8 :7 :6 :5 :2 :3 :4 :"
139Ø PRINT
140Ø PRINT "And reversed it a position 7 it would"
141Ø PRINT "look like this:"
142Ø PRINT
143Ø PRINT "Positions:"
144Ø PRINT "          :Ø :1 :2 :3 :4 :5 :6 :7 :8 :9 :"
145Ø PRINT "          :--:--:--:--:--:--:--:--:--:--:"
146Ø PRINT "          :Ø :1 :9 :8 :7 :6 :5 :4 :3 :2 :"
147Ø PRINT
148Ø GOSUB 1Ø9Ø
149Ø PRINT "}";
15ØØ POSITION 13,3
151Ø PRINT "*** REVERSER ***"
152Ø POSITION 2,7
153Ø PRINT "A final reversal at position 2 would"
154Ø PRINT "complete the list as this"
155Ø PRINT
156Ø PRINT "Positions:"
157Ø PRINT "          :Ø :1 :2 :3 :4 :5 :6 :7 :8 :9 :"
158Ø PRINT "          :--:--:--:--:--:--:--:--:--:--:"
159Ø PRINT "          :Ø :1 :2 :3 :4 :5 :6 :7 :8 :9 :"
16ØØ PRINT
161Ø PRINT "You win when the list is sorted in"
162Ø PRINT "ascending order as shown above."
163Ø PRINT
164Ø PRINT "Good luck !!!"
165Ø GOSUB 1Ø9Ø
166Ø FOR I=Ø TO 9
167Ø L1(I)=I
168Ø NEXT I
169Ø FOR I=Ø TO 9
17ØØ X=INT(RND(1)*1Ø)
171Ø T=L1(I)
172Ø L1(I)=L1(X)
173Ø L1(X)=T
174Ø NEXT I
```



```
1750 PRINT " esc clear ";
1760 POSITION 13,3
1770 PRINT "*** REVERSER ***"
1780 POSITION 2,7
1790 PRINT "Positions:"
1800 PRINT "          :Ø :1 :2 :3 :4 :5 :6 :7 :8 :9 :"
1810 PRINT "          :--:--:--:--:--:--:--:--:--:--:"
1820 PRINT "          :Ø :1 :2 :3 :4 :5 :6 :7 :8 :9 :"
1830 FOR I=Ø TO 9
1840 POSITION I*3+9,1Ø
1850 PRINT L1(I)
1860 NEXT I
1870 PRINT
1880 PRINT "Tries :";TR
1890 A=Ø
1900 FOR I=Ø TO 9
1910 A=A+(L1(I)=I)
1920 NEXT I
1930 IF A=1Ø THEN 2Ø7Ø
1940 PRINT
1950 PRINT "Reverse at which position ";
1960 INPUT A$
1970 A=INT(VAL(A$))
1980 IF A<Ø OR A>9 THEN 183Ø
1990 MDL=INT((9-A)/2)
2000 FOR I=A TO A+MDL
2010 T=L1(I)
2020 L1(I)=L1(9+A-I)
2030 L1(9+A-I)=T
2040 NEXT I
2050 TR=TR+1
2060 GOTO 183Ø
2070 POSITION 2,16
2080 PRINT "YOU GOT IT !!!"
2090 PRINT "It took you ";TR;" tries."
2100 PRINT "Would you like to play again Y/N ";
2110 INPUT A$
2120 IF LEN(A$) THEN IF A$="Y" THEN 166Ø
2130 END
```

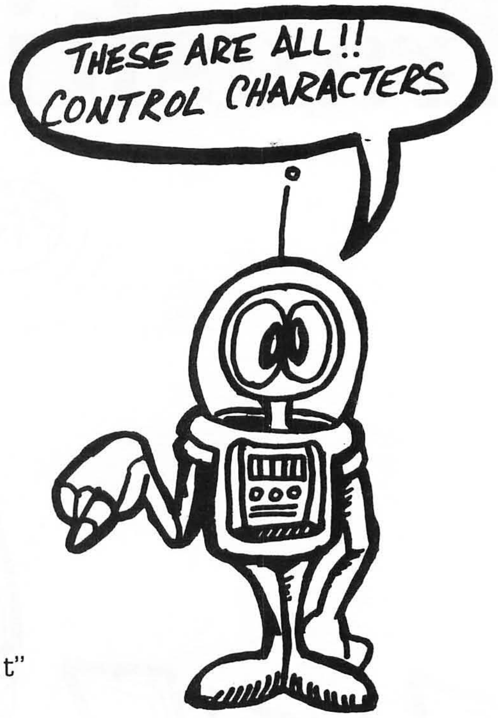


Transition

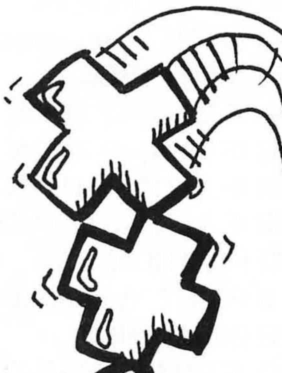


Here is a game that is guaranteed to wrinkle your brow. All the action takes place in a row of nine boxes. The four "+" (plus) signs on the right have to trade places with the four "–" (minus) signs on the left. There are at least 511 wrong combinations of moves assuming you don't repeat youself, and only one correct combination. The game board is drawn in lines 1640 to 1680. You'll see the borders of the game board squares appear on the Atari screen when you enter the control characters listed in the book. Lines 1060 to 2050 reads the keyboard and gives a value to A that corresponds to the 1 to 9 keys on the Atari. This is faster than asking for an input that requires a <RETURN>. In lines 1770 to 1830 a FOR NEXT loop sets flag F to 1 if the the game is not solved. There is no test for "hung up," but you will know when you are stuck and will voluntarily give up.

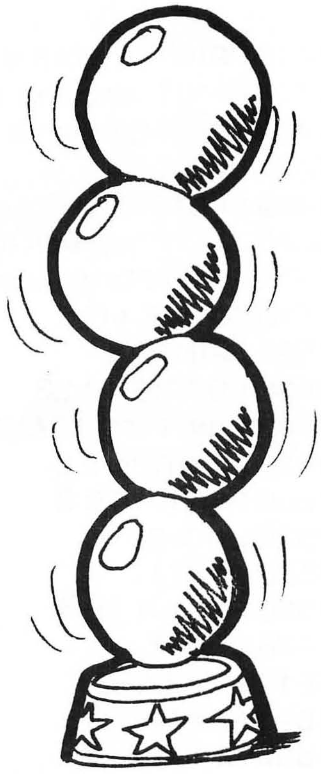
```
1000 REM  ***********************************
1010 REM *                                   *
1020 REM *           TRANSITION              *
1030 REM *                                   *
1040 REM  ***********************************
1050 REM
1060 REM DIMENSION VARIABLES
1070 DIM A$(20),L1(10)
1080 REM
1090 REM SKIP SUBROUTINES
1100 GOTO 1150
1110 POSITION 2,23
1120 PRINT "Press 'RETURN' to continue ";
1130 INPUT A$
1140 RETURN
1150 GRAPHICS 0
1160 T=0
1170 POSITION 11,3
1180 PRINT "*** TRANSITION ***"
1190 POSITION 2,6
1200 PRINT "The game of Transition will present"
1210 PRINT "you with a list of nine digits. The"
1220 PRINT "list will look like this:"
1230 PRINT: REM DRAW ROWS OF BOXES
1240 PRINT "        [R][W][R][W][R][W][R][W][R][W][R][W][R][W][R][W][R][E]"
1250 PRINT "COLUMN :1:2:3:4:5:6:7:8:9:"
1260 PRINT "        [A][R][R][R][R][R][R][R][R][R][D]"
1270 PRINT "        :[T]:[T]:[T]:[T]: : : : : :"
1280 PRINT "        [Z][R][X][R][X][R][X][R][X][R][X][R][X][R][X][R][X][R][C]"
1290 PRINT
1300 PRINT "The object of the game is to"
1310 PRINT "transpose the original character"
1320 PRINT "positions. Try to reverse the"
1330 PRINT "dots ([T]) and the plus signs () into"
1340 PRINT "one another's previous positions."
1350 GOSUB 1110
1360 PRINT "}";
1370 POSITION 11,3
1380 PRINT "*** TRANSITION ***"
1390 POSITION 2,6
1400 PRINT "The '[T]' character can only move to"
1410 PRINT "the right, and the " character can"
1420 PRINT "only move to the left."
1430 PRINT
```



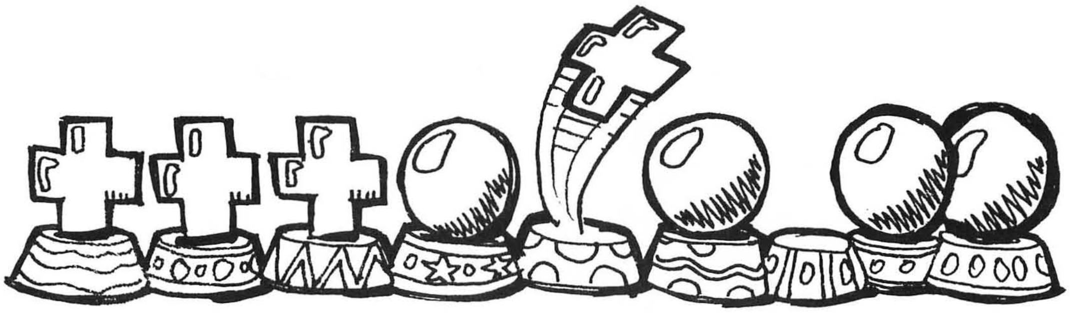
```
144Ø PRINT "A move is made by moving to an empty"
145Ø PRINT "space, or by jumping over an opposing"
146Ø PRINT "piece into an empty space."
147Ø PRINT
148Ø PRINT "To make a move, you enter the"
149Ø PRINT "position number of the moving piece."
15ØØ PRINT "To quit, enter zero (Ø)."
151Ø GOSUB 111Ø
152Ø PRINT "}";
153Ø POSITION 11,3
154Ø PRINT "*** TRANSITION ***"
155Ø POSITION 2,6
156Ø FOR I=1 TO 9
157Ø L1(I)=Ø
158Ø NEXT I
159Ø FOR I=1 TO 4
16ØØ L1(I)=1
161Ø L1(1Ø-I)=2
162Ø NEXT I
163Ø NM=Ø
164Ø PRINT "          [R][W][R][W][R][W][R][W][R][W][R][W][R][W][R][W][R][E]"
165Ø PRINT "COLUMN :1:2:3:4:5:6:7:8:9:"
166Ø PRINT "          [A][R][R][R][R][R][R][R][R][D]"
167Ø PRINT "          :[T]:[T]:[T]:[T]: :::::"
168Ø PRINT "          [Z][R][X][R][X][R][X][R][X][R][X][R][X][R][X][R][X][R][C]"
169Ø POKE 752,1
17ØØ PRINT
171Ø FOR I=1 TO 9
172Ø POSITION 2*I+9,9
173Ø IF L1(I)=1 THEN PRINT "[T]";
174Ø IF L1(I)=2 THEN PRINT "";
175Ø IF L1(I)=Ø THEN PRINT " ";
176Ø NEXT I
177Ø F=Ø
178Ø IF L1(5)< >Ø THEN F=1
179Ø FOR I=1 TO 9
18ØØ IF I<5 THEN IF L1(I)=1 THEN F=1
181Ø IF I>5 THEN IF L1(I)=2 THEN F=1
182Ø NEXT I
183Ø IF F=Ø THEN 232Ø
184Ø POSITION 2,12
185Ø PRINT "Move (Ø-9) ?       [T] esc ctl right arrow "
186Ø POSITION 14,12
187Ø B=PEEK(764)
```

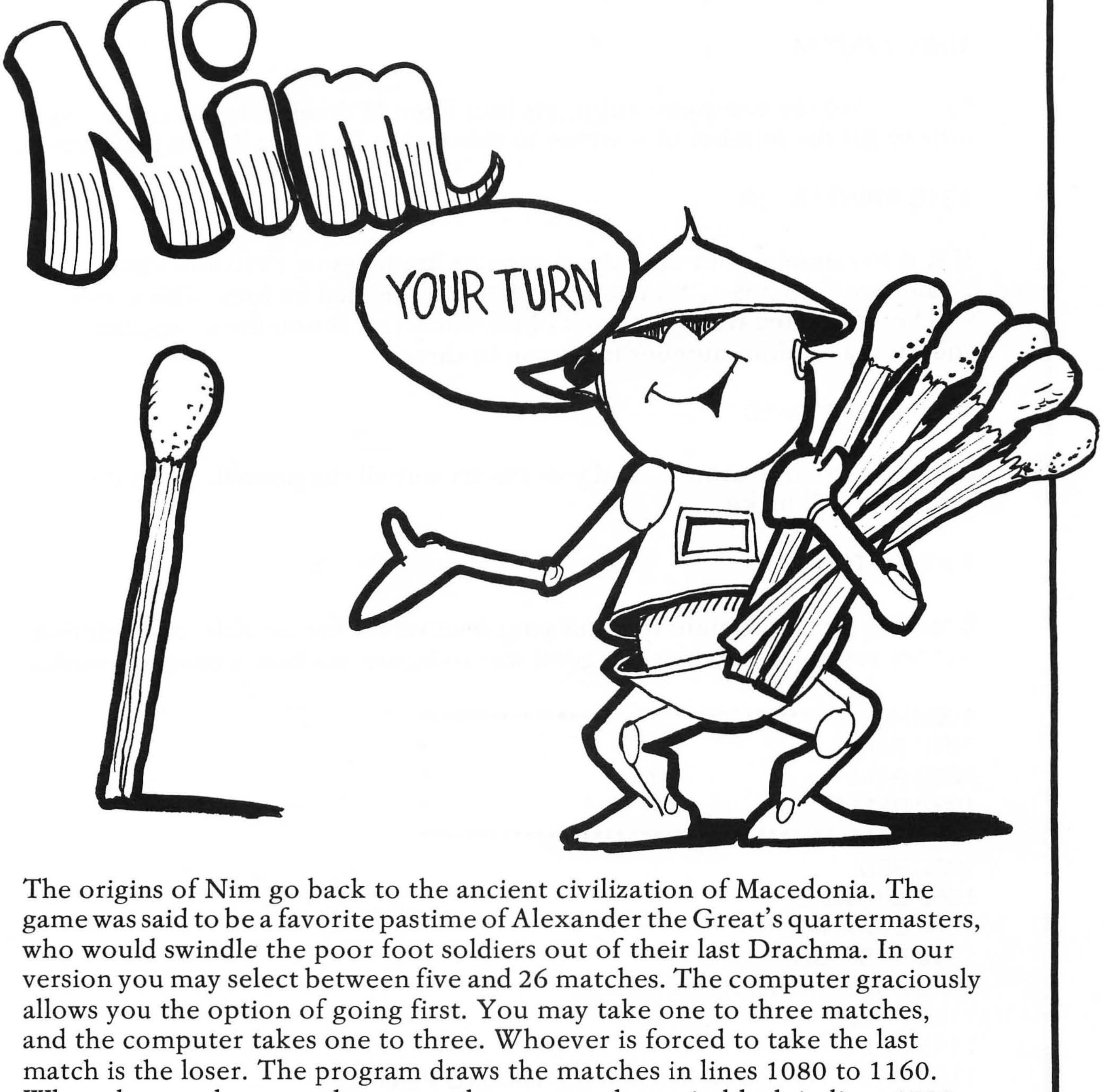


```
1880 A=-1
1890 T=T+1
1900 IF T<6 THEN 1960
1910 T=0
1920 BLNK=ABS(1-BLNK)
1930 IF BLNK THEN PRINT " "
1940 IF BLNK=0 THEN PRINT " ";
1950 POSITION 14,12
1960 IF B=50 THEN A=0
1970 IF B=31 THEN A=1
1980 IF B=30 THEN A=2
1990 IF B=26 THEN A=3
2000 IF B=24 THEN A=4
2010 IF B=29 THEN A=5
2020 IF B=27 THEN A=6
2030 IF B=51 THEN A=7
2040 IF B=53 THEN A=8
2050 IF B=48 THEN A=9
2060 IF A=-1 THEN 1870
2070 POKE 764,255
2080 PRINT A;
2090 POSITION 2,14
2100 PRINT "                ";
2110 IF A=0 THEN 2320
2120 IF L1(A)>0 THEN 2160
2130 POSITION 2,14
2140 PRINT "That space is empty} ";
2150 GOTO 1860
2160 IF L1(A)=1 THEN D1=1
2170 IF L1(A)=2 THEN D1=-1
2180 IF A+D1=0 THEN 2200
2190 IF A+D1<10 AND A+D1>0 THEN 2230
2200 POSITION 2,14
2210 PRINT "It cannot move further}";
2220 GOTO 1860
2230 IF L1(A+D1)>0 THEN 2260
2240 L1(A+D1)=L1(A)
2250 GOTO 2290
2260 IF A+D1+D1>9 OR A+D1+D1<0 THEN 2200
2270 IF (L1(A+D1)=L1(A)) OR (L1(A+D1+D1)>0) THEN 2200
2280 L1(A+D1+D1)=L1(A)
2290 L1(A)=0
2300 NM=NM+1
2310 GOTO 1710
```



```
2320 REM GAME OVER
2330 POKE 752,0
2340 POSITION 2,14
2350 PRINT "The game is over !!! "
2360 IF F=0 THEN 2400
2370 PRINT "You got stuck after ";NM;" moves."
2380 PRINT "Better luck next time."
2390 GOTO 2420
2400 PRINT "You did it !!!"
2410 PRINT "And it only took you ";NM;" moves."
2420 PRINT
2430 PRINT "Would you like to play again Y/N ";
2440 INPUT A$
2450 IF LEN(A$) THEN IF A$(1,1)="Y" THEN 1520
2460 END
```





The origins of Nim go back to the ancient civilization of Macedonia. The game was said to be a favorite pastime of Alexander the Great's quartermasters, who would swindle the poor foot soldiers out of their last Drachma. In our version you may select between five and 26 matches. The computer graciously allows you the option of going first. You may take one to three matches, and the computer takes one to three. Whoever is forced to take the last match is the loser. The program draws the matches in lines 1080 to 1160. When the matches are taken away they are un-drawn in black in lines 1530 to 1610. Change the color numbers to see which command draws which parts. Lines 1310 to 1360 is the computer's fiendish move. At the beginning of the computer's move, the matches remaining are in the variable M. To see how the program works put a GOTO at the beginning of the program to bypass everything but the computer's calculation.

```
10 GOTO 1305
```

Now put in your own value for M.

```
1305 INPUT M
```

In line 1310 the computer subtracts four from M divides the remainder by four to get the number of matches to subtract in R. Add a line to print out R.

```
1315 PRINT "R=";R
```

If R is any number but one, the computer branches to 1350 and figures out a value for C that is a "remainder" when R is divided by four. This C value will be subtracted from the stack of matches. If R is one the computer subtracts a random number from one to three.

```
1355 PRINT "C=";C
```

Add a line to loop around, and you can try out all the possible plays the computer will make.

```
1365 GOTO 1305
```

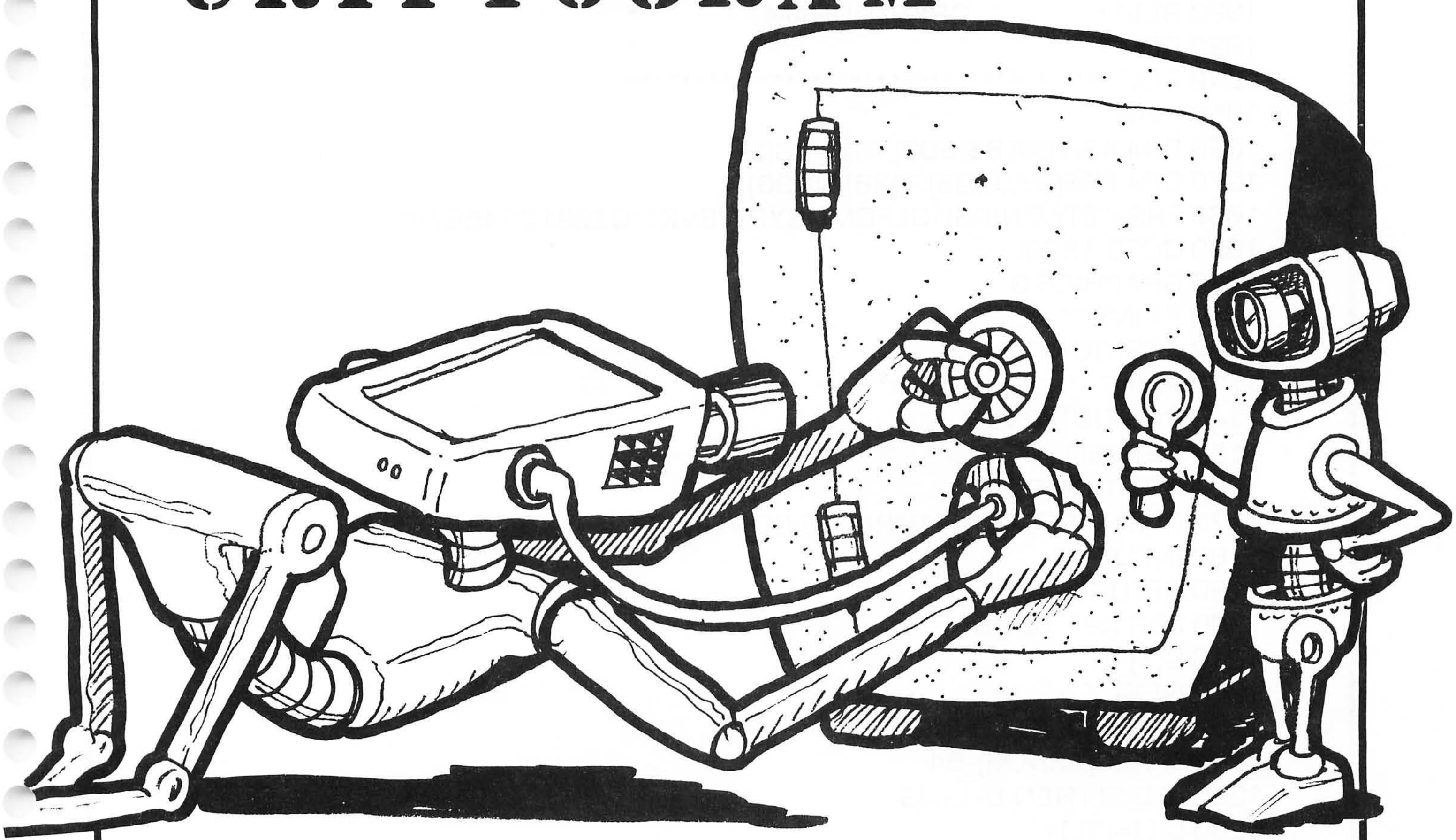
Breaking into a program to enter your own values for variable, and printing out the resulting variables is a good way to figure out how a program works.

```
1000 REM  ***********************************
1010 REM *                                 *
1020 REM *               NIM               *
1030 REM *                                 *
1040 REM  ***********************************
1050 REM
1060 DIM A$(2)
1070 GRAPHICS 0
1080 POSITION 14,3
1090 PRINT "*** NIM ***"
1100 PRINT
1110 PRINT
1120 PRINT "THIS IS THE GAME OF NIM. TO PLAY"
1130 PRINT "YOU ENTER THE NUMBER OF MATCHES TO"
1140 PRINT "REMOVE. THE COMPUTER WILL THEN MAKE"
1150 PRINT "IT'S MOVE. THE ONE WHO TAKES THE"
1160 PRINT "LAST MATCH LOSES."
1170 POSITION 2,23
```

```
1180 PRINT "PRESS 'RETURN' TO CONTINUE.";
1190 INPUT A$
1200 PRINT "}"
1210 POSITION 14,3
1220 PRINT "*** NIM ***"
1230 PRINT :PRINT
1240 ? "HOW MANY MATCHES? ( 4 TO 26)"
1250 INPUT A$:IF A$="" THEN A$="0"
1260 M=VAL(A$)
1270 IF M<4 OR M>26 THEN 1250
1280 GRAPHICS 5
1290 L=0
1300 N=M
1310 W=M
1320 SETCOLOR 0,3,6
1330 SETCOLOR 1,1,10
1340 FOR I=1 TO M
1350 COLOR 2
1360 PLOT 3*I,5
1370 DRAWTO 3*I,15
1380 COLOR 1
1390 PLOT 3*I,4
1400 NEXT I
1410 ? "DO YOU WANT TO MOVE FIRST?"
1420 INPUT A$
1430 IF A$(1,1)="N" THEN 1540
1440 ? "THERE ARE NOW ";M;" MATCH(ES)."
1450 ? "TAKE 1 TO 3 MATCHES ";
1460 INPUT A$:IF A$="" THEN A$="0"
1470 H=VAL(A$)
1480 IF H>3 OR H<1 OR H>M THEN PRINT "BAD NUMBER.
     TRY AGAIN.":GOTO 1440
1490 X=L+H
1500 GOSUB 1770
1510 L=X
1520 M=M-H
1530 IF M=0 THEN 1690
1540 REM
1550 IF M=1 THEN 1750
1560 R=M-4*INT(M/4)
1570 IF R< >1 THEN 1600
1580 C=INT(3*RND(1))+1
1590 GOTO 1610
1600 C=(R+3)-4*INT((R+3)/4)
```

```
1610 X=L+C
1620 ? "MY TURN, I'M THINKING"
1630 GOSUB 1770
1640 L=X
1650 M=M-C:IF M=0 THEN 1750
1660 GOSUB 1770
1670 ? "I TOOK ";C;" MATCH(ES)"
1680 GOTO 1440
1690 REM
1700 ? "I WON! BETTER LUCK NEXT TIME"
1710 ? "DO YOU WANT TO TRY AGAIN "
1720 INPUT A$
1730 IF A$(1,1)< >"N" THEN 1240
1740 PRINT "THANKS FOR THE GAME":END
1750 ? "YOU WON!!! NICE GOING":GOTO 1710
1760 ?
1770 REM
1780 COLOR 0
1790 FOR I=L TO X
1800 FOR J=1 TO 12
1810 PLOT 3*I,3+J
1820 NEXT J
1830 FOR K=1 TO 10
1840 NEXT K
1850 NEXT I
1860 RETURN
```

CRYPTOGRAM

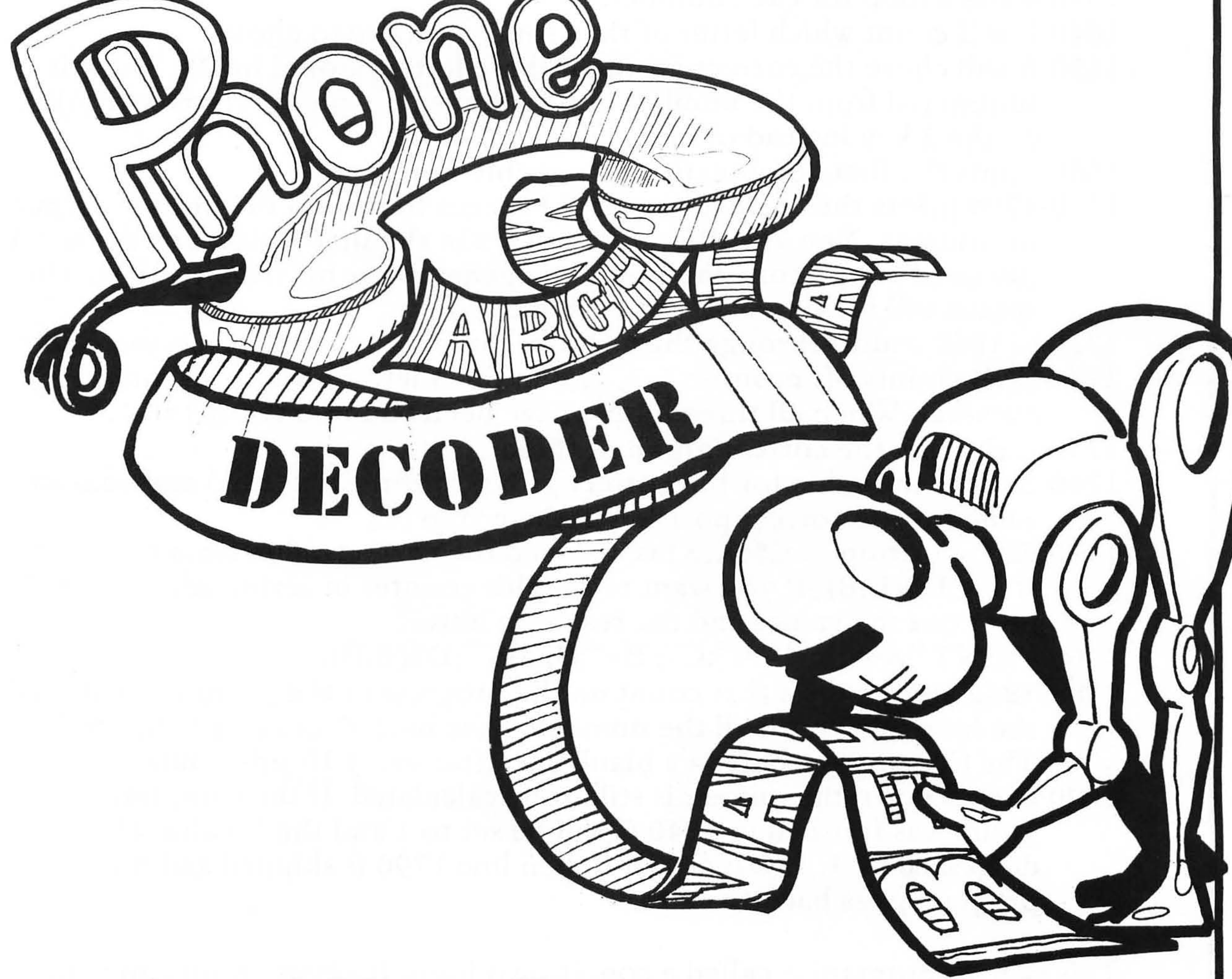


During World War II American scientists were working on large "number crunching" machines to do ballistics calculations. These machines evolved into the computers of today. At the same time the British were working on machines to code and decode secret messages that are a key part of military operations. Their work made a vital contribution to the development of programming. The underlying principle in decoding is the frequency table that lists the alphabet in the order of the frequency that the letters appear in English text. This alphabet is stored in our decoder program in line 1080. Frequency tables have been constructed for other languages. One of the easiest ways to break a code is to dupe the enemy into sending a known message in their code; this was called a "Trojan Horse." Double agents would leak fake messages to the enemy in hopes that enemy radio operators would send these known messages in code. The code our program works with is a straight substitution type. The most likely answer is generated first, then you are invited to tinker with additional substitutions suggested by a few letters that string together in sensible patterns. The longer the coded message is, the better chance a few substitutions will generate coherent words.

```
1ØØØ REM  ************************************
1Ø1Ø REM *                                  *
1Ø2Ø REM *          CRYPTOGRAM              *
1Ø3Ø REM *                                  *
1Ø4Ø REM  ************************************
1Ø5Ø REM
1Ø6Ø DIM A$(2ØØ),B$(2ØØ),T$(1),Y$(2)
1Ø7Ø DIM FR$(36),C(36),R(36),S$(36)
1Ø8Ø FR$="ETAONIRSHDLFCMUGYPWBVKXIQZØ912345678"
1Ø9Ø GOTO 153Ø
11ØØ GRAPHICS Ø
111Ø PRINT "}"
112Ø POSITION 1Ø,4
113Ø PRINT "*** DECODE ***"
114Ø POSITION 2,7
115Ø RETURN
116Ø POSITION 2,23
117Ø PRINT "PRESS 'RETURN' TO CONTINUE";
118Ø INPUT A$
119Ø RETURN
12ØØ FOR X=1 TO 36
121Ø C(X)=Ø
122Ø NEXT X
123Ø FOR X=1 TO LEN(A$)
124Ø L=ASC(A$(X,X))-64
125Ø IF L<Ø THEN L=L+43
126Ø C(L)=C(L)+1
127Ø NEXT X
128Ø S$="ABCDEFGHIJKLMNOPQRSTUVWXYZØ123456789"
129Ø F=Ø
13ØØ FOR X=1 TO 35
131Ø IF C(X)>=C(X+1) THEN 138Ø
132Ø T=C(X):C(X)=C(X+1)
133Ø C(X+1)=T
134Ø T$=S$(X,X)
135Ø S$(X,X)=S$(X+1,X+1)
136Ø S$(X+1,X+1)=T$
137Ø F=1
138Ø NEXT X
139Ø IF F=1 THEN 129Ø
14ØØ T=Ø
141Ø FOR X=1 TO 36
142Ø IF C(X)=Ø AND T=Ø THEN T=X-1
143Ø NEXT X
```

```
1440 S$=S$(1,T)
1450 REM
1460 B$=A$
1470 FOR X=1 TO LEN(A$)
1480 FOR Y=1 TO LEN(S$)
1490 IF A$(X,X)=S$(Y,Y) THEN B$(X,X)=FR$(Y,Y)
1500 NEXT Y
1510 NEXT X
1520 RETURN
1530 GOSUB 1100
1540 PRINT
1550 PRINT " ENTER A CRYPTOGRAM AND I WILL HELP"
1560 PRINT "YOU SOLVE IT. I KNOW THE MOST"
1570 PRINT "COMMONLY USED LETTERS AND THEIR ORDER"
1580 PRINT "WITH THIS INFORMATION AND SOMEONE"
1590 PRINT "LIKE YOURSELF TO HELP ME WE CAN SOLVE"
1600 PRINT "ALMOST ANY CRYPTOGRAM."
1610 PRINT
1620 PRINT "FIRST YOU TYPE IN THE CRYPTOGRAM."
1630 PRINT "I WILL PERFORM A COUNT AND REPLACE"
1640 PRINT "THE LETTERS WITH THEIR POSSIBLE REAL"
1650 PRINT "LETTER."
1660 GOSUB 1160
1670 GOSUB 1110
1680 PRINT "YOU CAN SUGGEST PROPER "
1690 PRINT "REPLACEMENTS ONE AT A TIME BY TYPING"
1700 PRINT "THEM IN WHEN I ASK. FOR EXAMPLE IF"
1710 PRINT "THE CODE WAS WIG AND THE MESSAGE WE"
1720 PRINT "HAD DECODED SO FAR WAS 'TUE' YOU"
1730 PRINT "COULD TYPE IH MEANING REPLACE THE I"
1740 PRINT "WITH H. AND WE WOULD GET THE CORRECT"
1750 PRINT "ANSWER."
1760 GOSUB 1160
1770 GOSUB 1110
1780 PRINT "INPUT YOUR CRYPTOGRAM 200 LETTERS MAX"
1790 PRINT "WITHOUT SPACES OR SPECIAL CHARACTERS"
1800 PRINT
1810 PRINT "ENTER: ";
1820 INPUT A$
1830 GOSUB 1200
1840 GOSUB 1110
1850 PRINT
1860 PRINT A$
1870 PRINT B$
```

```
1880 PRINT
1890 PRINT "REPLACE AB ";
1900 INPUT Y$
1910 IF Y$="" THEN END
1920 T$=Y$(1,1):Y$=Y$(2,2)
1930 F=0
1940 FOR X=1 TO LEN(S$)
1950 IF S$(X,X)=T$ THEN FR$(X,X)=Y$:F=1
1960 NEXT X
1970 IF F=0 THEN 2000
1980 S$(X,X)=T$
1990 FR$(X,X)=Y$
2000 GOSUB 1460:GOTO 1840
```



You may have watched the movie "War Games" and wondered how the computer could try every possible combination of phone numbers. This is a trivial task for a computer. There are fiendish advertisers who own computers that do nothing but phone through all possible combinations of phone numbers to play recorded messages to innocent persons who probably are eating dinner or taking a shower. We will not encourage any such anti-social behavior, but the following program will generate a list of all possible combinations of letters that can be made out of your phone number. You might have some easy-to-remember combination that you can tell people so they will never forget you. "Hi my name is Joe and my phone number is TREETOP." One slight catch, if your number has a 0 or 1 the program cannot use the number, because these numbers do not have equivalent letters. You could modify the program to use a Z for 0, but the number of words with Z's in them does not merit the effort. Lines 1340 to 1520 format and error trap your input. 1530 to 1590 opens the printer to I/O #1.

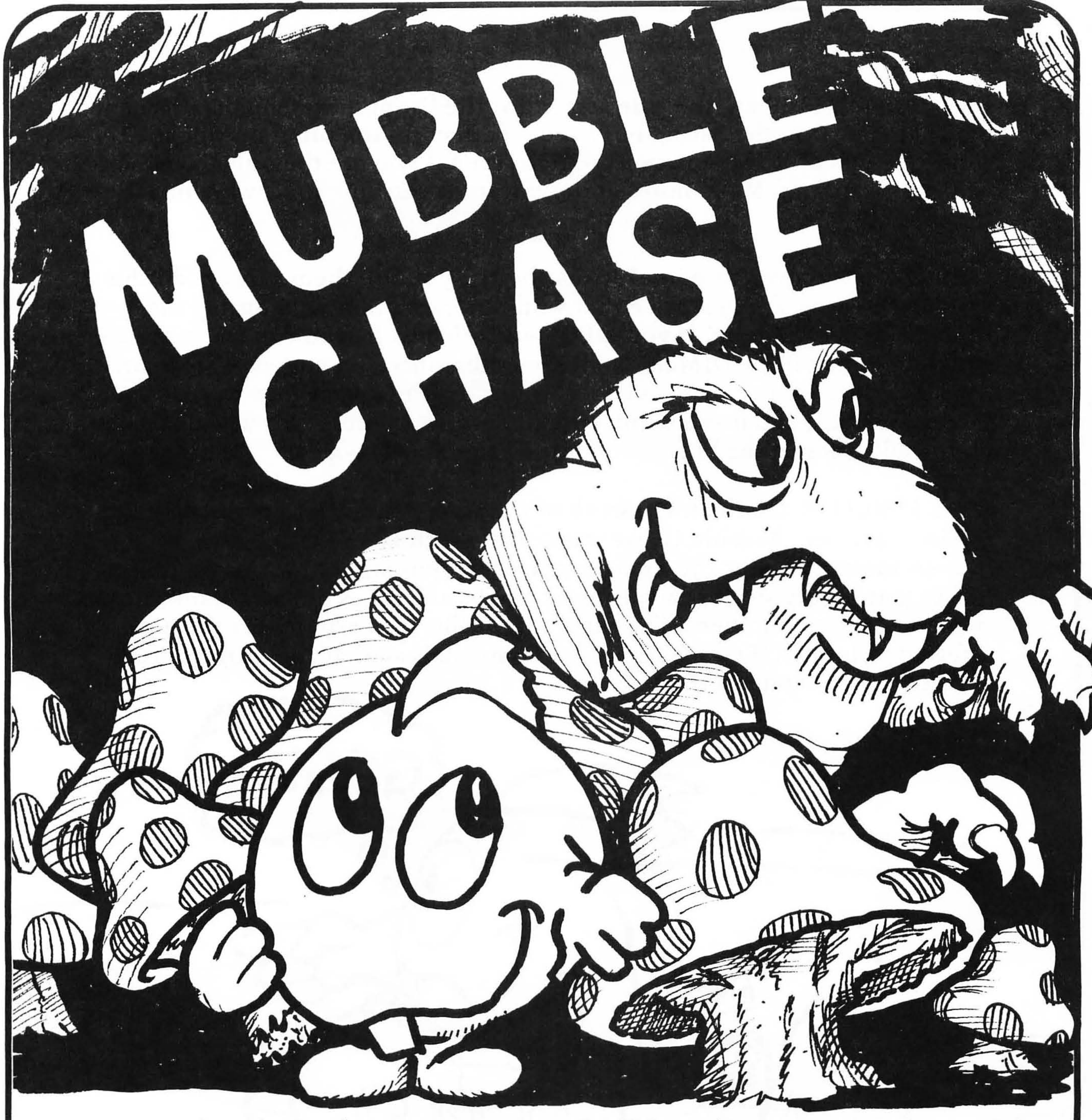
1610 N is set to the length of the phone number.
1630 starts a loop for each number.
1640 C will count which letter of the groups of three to chose.
1650 A will chose the correct group of three letters stored in D$. The 2 is subtracted from the number in B$ because the phone starts with ABC on the 2 key instead of on 0.
1660 prints the first combination of possible letters.
1680-1700 prints the correct number of spaces to line up the printer output in columns. You are allowed to put 0's in the first 3 places or the last 4 places of your phone mumbers and generate words from the rest. The spaces will fill where you put 0's in the input.
1720 to 1840 counts through the possible combinations and prints the results.
1740 C(X) counts off position 1,2, or 3 in the 3 letter groups for each numeral. When all three letters have been counted C is set to 1.
1750 A chooses the correct group of three.
1760 B takes the value for the correct group of three letter (A) and adds the value for the correct position in the group (C)
1770 Each position of C$(X,X) is assigned the correct value from alphabet string D$(B,B). If you want to see this counter in action add a line to print out the values and the resulting letter.
1765 PRINT "A=";A;" C=";C"; B=";B;" IS ";D$(B,B).
1780-1840 are the tests that count up the progress of the printouts and send the loop back until all the numbers have been decoded and printed. The CN counter puts in a blank line after every 10 print-outs.
1790 counts down the numerals still to be calculated. If the three letter group was finished in 1740 C will be set to 1 and the X value will be decreased by 1. If C is still at 0 then line 1790 is skipped and the program goes back to 1740.

This type of program is called a conditional loop. It keeps on printing out until every combination has been tried.

```
1000 REM ************************************
1010 REM *                                  *
1020 REM *          PHONE DECODER           *
1030 REM *                                  *
1040 REM ************************************
1050 REM
1060 DIM A$(10),B$(10),C$(7),D$(24),C(10),SP$(10)
1070 D$="ABCDEFGHIJKLMNOPRSTUVWXY"
1080 REM
1090 GOTO 1200
1100 GRAPHICS 0
1110 PRINT "}"
1120 POSITION 10,3
```

```
1130 PRINT "*** PHONE DECODER ***"
1140 POSITION 2,7
1150 RETURN
1160 POSITION 2,23
1170 PRINT "PRESS 'RETURN' TO CONTINUE";
1180 INPUT A$
1190 RETURN
1200 GOSUB 1100
1210 PRINT " THIS PROGRAM WILL TAKE A 7 DIGIT"
1220 PRINT "PHONE NUMBER IN THE FORM NNN-NNNN"
1230 PRINT "AND CONVERT IT TO ALL OF THE POSSIBLE"
1240 PRINT "ALPHA COMBINATIONS FOR YOUR"
1250 PRINT "INSPECTION."
1260 PRINT
1270 PRINT " UNFORTUNATLY SINCE THE Ø AND THE 1"
1280 PRINT "IN PHONE NUMBERS DO NOT HAVE ALPHA"
1290 PRINT "EQUIVALENTS PHONE NUMBER ELEMENTS"
1300 PRINT "WITH THOSE NUMBERS CANNOT BE DECODED."
1310 GOSUB 1160
1320 GOSUB 1110
1330 PRINT
1340 PRINT "ENTER A PHONE NUMBER ";
1350 INPUT A$
1360 IF A$="" THEN END
1370 PRINT
1380 IF LEN(A$)< >8 THEN 1320
1390 F=Ø
1400 FOR X=1 TO 3
1410 IF VAL(A$(X,X))<2 THEN F=1
1420 NEXT X
1430 B$=""
1440 IF F=1 THEN 1460
1450 B$=A$(1,3)
1460 F=Ø
1470 FOR X=5 TO 8
1480 IF VAL(A$(X,X))<2 THEN F=1
1490 NEXT X
1500 IF F=1 THEN 1520
1510 B$(LEN(B$)+1,LEN(B$)+5)=A$(5,8)
1520 IF B$="" THEN PRINT "CAN'T DO THAT ONE":GOTO 1330
1530 CLOSE #1
1540 PRINT
1550 PRINT "PRINTER (P:) OR SCREEN (E:) ";
1560 INPUT A$
```

```
157Ø IF A$< >"P:" THEN A$="E:"
158Ø REM
159Ø OPEN #1,8,Ø,A$
16ØØ PRINT #1;"COMBINATIONS FOR ";B$
161Ø N=LEN(B$)
162Ø C$=""
163Ø FOR X=1 TO N
164Ø C(X)=Ø
165Ø A=VAL(B$(X,X))-2
166Ø C$(X,X)=D$(A*3+1,A*3+1)
167Ø NEXT X
168Ø IF LEN(B$)=4 THEN SP$="  "
169Ø IF LEN(B$)=3 THEN SP$="  "
17ØØ IF LEN(B$)=7 THEN SP$=" "
171Ø CN=Ø
172Ø PRINT #1;C$;SP$;
173Ø X=N:F=Ø:C=Ø
174Ø C(X)=C(X)+1:IF C(X)=3 THEN C(X)=Ø:C=1
175Ø A=VAL(B$(X,X))-2
176Ø B=A*3+1+C(X)
177Ø C$(X,X)=D$(B,B)
178Ø IF C=Ø THEN 182Ø
179Ø X=X-1
18ØØ C=Ø
181Ø IF X>Ø THEN 174Ø
182Ø CN=CN+1:IF CN=1Ø THEN PRINT #1:CN=Ø
183Ø IF X>Ø THEN 172Ø
184Ø PRINT #1
```



Games of catch-me-if-you-can written in BASIC tend to exaggerate the leisurely pace of higher level languages. Mubble Chase is a snail's paced version of SNACK ATTACK™ or PAC MAN™, but there is a lot to be learned about graphics, loops, grids, color and arrays with this 'granddaddy' of the arcades. If you spend some time understanding this game, you will be able to program lots of graphic effects in your own games. For this reason, Mubble Chase was chosen for a lengthy description of all the BASIC statements.

Lines 1000 to 1100 are the title screen for the listings. Statements preceeded by REM appear in the listing but have no effect on the program when it runs. Use lots of REMS in your program even if you are the only one who will ever see them. Make notes about different versions and always include the date.

1120 DIMension variables and strings before you use them. Always do this early in the program in a place that won't be executed again during the running of the program. If you DIM a variable just before you use it there is a possibility it will be DIMenioned more than once and generate an error. Atari's BASIC editor makes it easy to add variables and strings to an earlier DIM statement, and leaving lots of spaces in your numbering will also allow you to add lines to the beginning of the program if needed.

Lines 1130,1170 are the framework of the structured program discussed in the introduction. Mubble Chase was written with all the GOSUBs numbered in even thousands, but the final program was renumbered with a renumber utility that evenly spaces the program by 10s and takes out any blanks in the sequence of line numbers. The renumber utility also renumbers the lines referred to in the GOSUBs and other commands so that the program runs correctly after renumbering.

1220-1320 is the first text screen of instructions.

1330 sends the program to the clear screen subroutine.

1340-1460 is a second screen full of instructions. 1340 starts with the clear screen command. In order to put a control function into your program it must first be in a print statement and be preceeded by [esc] when typed. On the Atari screen, the clear screen command appears as a slant-back arrow, and on most printers it comes out as a close curly bracket "}". A set of arrows and directions are drawn by lines 1420-1460. Getting the arrow to print on the screen requires three escapes in a row to be typed. The first [esc] instructs the computer to print the next control character into the listing instead of executing it directly. The next [esc] is then printed into the program line. The third [esc] instructs the computer to enter the next control character into the program line, and this character is the arrow key. The Atari screen will show a little EC character and the arrow in the print statement. If you only typed one [esc] the arrow would be in the program line, but when the program was run the arrows would be executed rather than printed and would move the cursor around just as it does in editing.

1490-1500 is a subroutine to hold the title text on your screen until you press RETURN. Subroutines are a handy device for performing the same action a number of times.

1480 RETURN's the program to the next line after the GOSUB in 1130. The program is then sent to 1510 for the setup section. Notice that the flow of the program goes around the subroutine lines 1490-1500. This subroutine is "nested" in a larger subroutine, so getting into these lines by mistake would get a RETURN from the wrong location and your program would soon be in the "Twilight Zone." If your program wasn't already in a subroutine and fell into one the computer would generate ERROR- 16 AT LINE X -- RETURN without GOSUB.

1540 sets the graphics to mode 5 and adds 16 to fill in the text area at the bottom of the screen with graphics. This mode gives 80 columns by 48 rows and four colors.

1550 SETCOLOR is the command the sets up the values for the entire screen. COLOR sets the color to be plotted at a single point or line. The first value of SETCOLOR is the register number. Registers 0,1 and 2 hold the foreground colors and 4 is the register that holds the background color. The second value is the color and the third value is the luminance (brightness). You can experiment with other SETCOLOR commands and get a feel for the color possabilities of the Atari. Disconnect line 1550 from the program by using the [ctrl] and arrow keys to put the cursor in the space to the right of number 1550. Type three [ctrl][insert] to make a space and type REM in front of the line: 1550 REM SETCOLOR 0,3,9 Now use lines 1551-1559 to add other SETCOLOR combinations to experiment on.

1560-1610 draws an outline box in COLOR 1. After a COLOR has been plotted SETCOLOR will change all the parts of the sceen that contain that color.

1620-1640 sets a value to be used as the color for the Mubble Eaters in the play section.

1650-1730 draws the grid using FOR-NEXT, PLOT, and DRAWTO. The loop starts with 15 as the left border, 61 as the right border, and counts by 4. At each interval (I) a point is plotted at the top (I,3) and a line drawn from there to the bottom (I,42). Another FOR-NEXT loop draws the horizontal lines.

1740-1790 draws the food points at each intersection using nested FOR-NEXT loops. If you've never met a nested loop before there is a technique for demonstrating these loops. Change 1540 to GRAPHICS 5, this will leave a space at the bottom of the screen for text. Add these lines:

```
1772 PRINT "I=";I;" J=";J,
1774 FOR PAUSE=1 TO 200:NEXT PAUSE
```

The calculations of the food point locations will print out below the grid. You will be able to watch the points being drawn because the delay loop will slow down the operation of the program. Adding PRINT statements for variables is a good way to see how things work in the program. Change 1540 back to GRAPHICS 5+16 when you're done experimenting.

When the play section starts the three Mubble Eaters are drawn by lines 1850-1970. The position of the Mubble Eater is stored in a two-dimensional array. ME(1,1) is not a position, it is the name of a variable where the position is stored. Two of these variables, ME(1,1) and ME(1,2) are needed to plot one position. The variable ME(1,0) is not used until later in the program, where it is used to store the old position of ME while it is moving along.

Arrays work like rows of pigeon holes and are very useful, because the values for the arrays can be manipulated with variables as they are in line 1960. A loop if I=1 to 3 steps through the 3 Mubble Eater (ME) arrays:

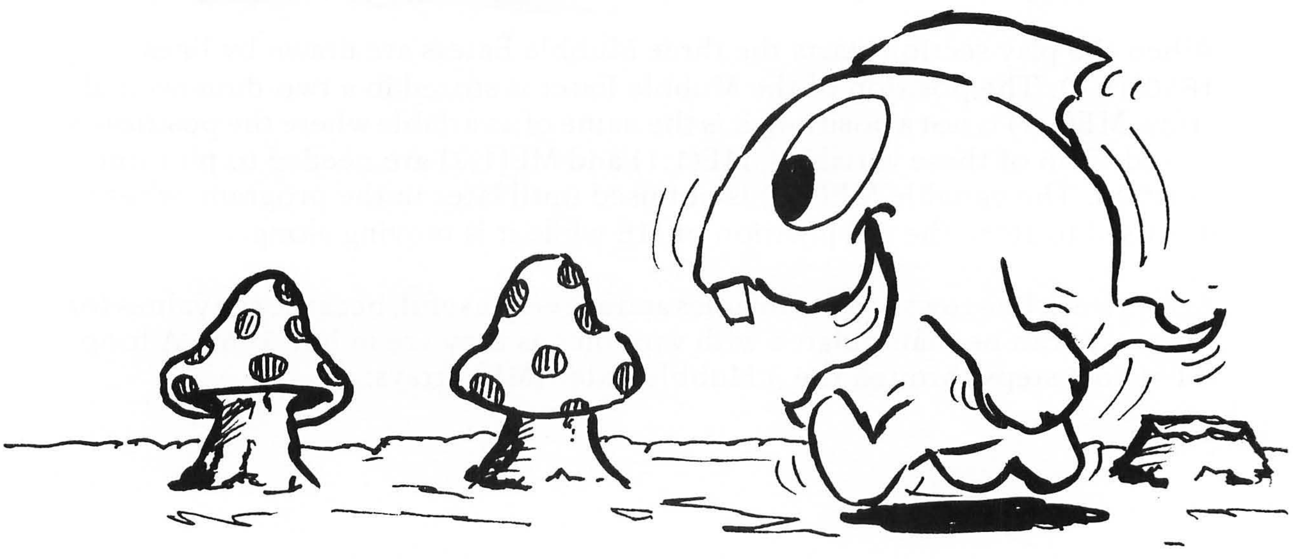
	COLUMN Ø	COLUMN 1	COLUMN 2
ROW 1	ME(1,Ø)=3	ME(1,1)=15	ME(1,2)=3
ROW 2	ME(2,Ø)=3	ME(2,1)=59	ME(2,2)=3
ROW 3	ME(3,Ø)=3	ME(3,1)=59	ME(3,2)=43

The rows are used for the three ME's and the columns store the grid positions. This array allows the loop to use I for the three ME's and read their grid positions as ME(I,1),ME(I,2).

A similar array is used for the Mubble, but instead of plotting three different single dots the Mubble is a little creature three dots longs. By means of some clever statements the Mubble is able to turn corners and keep his three dots together as he goes.

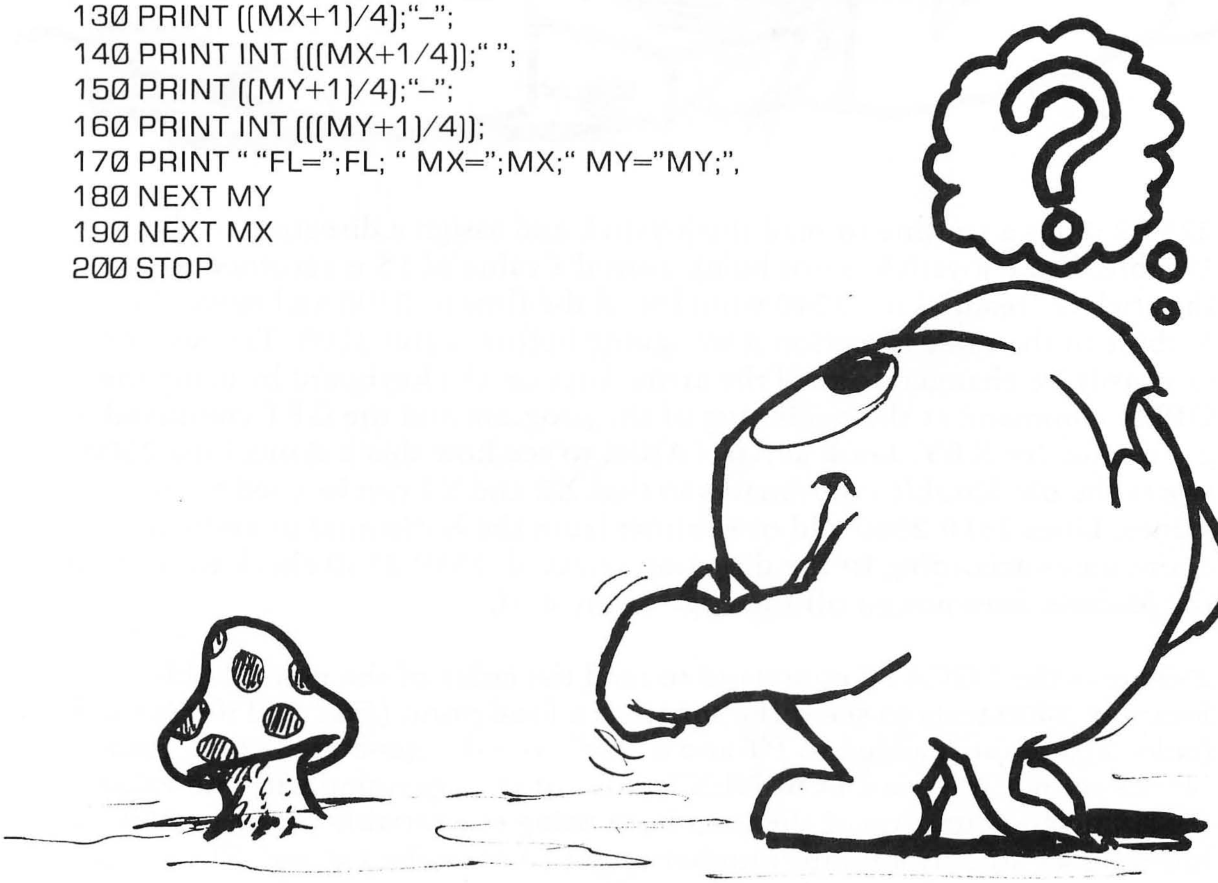
2080 to 2140 draws the Mubble's starting position in two different colors and two different sounds with brief pauses in between. The result is a square that flashes and beeps.

2160-2200 is the main play loop. First, the Mubble is moved by a subroutine starting at 2210. When the program returns the next line checks to see if the Mubble has eaten all the food points. If it has, the program is sent to 2830 which is the end routine. 2180 moves the Mubble Eater. If the ME has devoured the MU then the program will return from this subroutine with HI equal to 1. 2190 IF NOT HI THEN 2160 is a conditional test. As long as HI is 0, NOT HI will fullfill the IF condition and the part of the line after THEN will be executed. The play loop is a conditional loop with two tests; either the Mubble gets all the food points or the Mubble Eater gets him.



Line 2210 is a Boolian truth test to determine whether the Mubble is on an intersection of the grid and allowed to turn a corner or not. Before you put this line in your program put it on the "work bench," and make a loop and print statement to check it out. The workbench is a program area that can be used to test lines without interfering with the main program. The test program below will print all the combinations of values of the grid and show you which ones test out as valid intersections.

```
100 FOR MX = 15 TO 61
110 FOR MY = 3 TO 46
120 FL = ((MX+1)/4) = INT (((MX+1)/4)) AND ((MY+1)/4) = INT (((MY+1)/4))
130 PRINT ((MX+1)/4);"–";
140 PRINT INT (((MX+1/4));" ";
150 PRINT ((MY+1)/4);"–";
160 PRINT INT (((MY+1)/4));
170 PRINT " "FL=";FL; " MX=";MX;" MY="MY;",
180 NEXT MY
190 NEXT MX
200 STOP
```



When you run this program you will see that the left hand side of the equality test will be a decimal number until MX+1 is evenly divisable by 4. The same value preceded by INT will be rounded off, and will only match the even divisions. The AND statement requires that both MX+1 and MY+1 be evenly divisable by 4. FL will only equal 1 if MX+1 and MY+1 are the coordinates of the grid. When you are satisfied that line 120 is "doing its thing" then renumber it as 2210. Save the testing program and add a line 10 GOTO 1000 to run the game.

2230-2380 is a routine to read the joystick and assign a direction to the Mubble. If the joystick is not being moved a value of 15 is returned when the stick is "read." Line 2240 would send the flow to 2300 and move the Mubble in the same direction it was going before. 2260-2290. This section can easily be changed to read the arrow keys on the keyboard by using the OPEN command at the beginning of the program and the GET command to get a value for KEY. Look at Atari Artist to see how this is done. Line 2300 stores the old Mubble coordinates so that X2 and Y2 can be used for new values. Lines 2310-2380 add or subtract from the horizontal or vertical coordinates according to the direction selected. 2350-2380 check to see that the Mubble does not go off the edge of the grid.

2390 uses the LOCATE command to read the color of the new Mubble location. 2400 tests to see if the square is a food point (FD), and if it is food (color 2) a point is added to PT and a "zip" sound is generated. 2410 turns off the sound. We use a FOR-NEXT loop for the zip sound with the value of the sound and duration of the loop both using the variable (I). Since 250 is a low note we step by a minus number to get an ascending sound. The entire loop is in one line because it follows and IF statement and only sounds off in response to a food point.

2420-2530 moves the Mubble by erasing the tail position MU(1,1),MU(1,2) and then going through a series of swaps, middle becomes tail, head becomes middle and head becomes newly plotted X2 and Y2. This process works the same whether the Mubble is going straight or around a corner. 2540 RETURNs to the main play loop.

2550-2810 is a FOR-NEXT loop that moves the three Mubble Eaters in turn.

2560-2570 is the grid crossing test expained in line 2210, except that MX has to be changed to ME(I,1). We can't use ME(I,1) directly because line 2570 would be longer than Atari BASIC allows, so we define A as ME(I,1) and use A in the formula instead.

2590 -2600 are ON-GOTO statements with random numbers sending the program to different lines. Notice that the line numbers are repeated in line 2590, and there is a 50/50 chance of going to 2600 and only a 1 in 6 chance of going to line 2650.

Lines 2670-2700 compares the ME position to the Mubbles X and Y coordinates so the MEs will advance toward the Mubble.

2710-2740 checks to see that the Mubble Eaters stay within the grid.

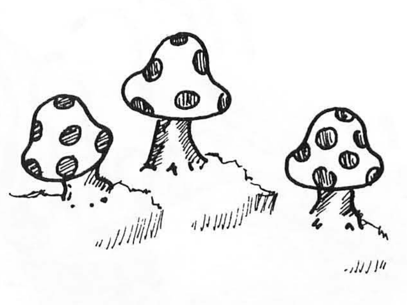
2750-2780 reads the color of the space the Mubble Eater moves to and stores it in SP(I). When the ME next gets plotted this color is placed in the space the ME vacated. This allows the ME to walk over a food point and put it back down where it belongs without eating it.

2790-2800 tests to see if the Mubble and Mubble Eater are colliding. The values have to be translated to single letters to get all the tests on one line within the allowable limit of line length.

2820 RETURNs to the play loop where the game either continues or goes to the end routine.

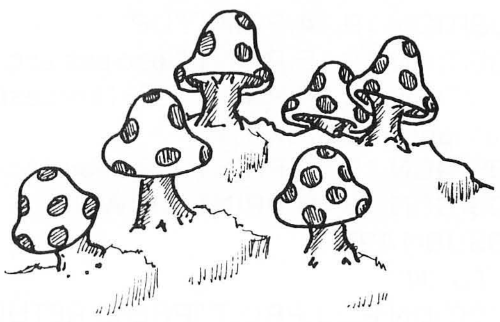
2830-3030 is the end routine. The perfect score results in a plain text screen which you are invited to improve upon. The destruction of the Mubble will result in a recap of the points devoured in large text using GRAPHICS 17, PRINT #6 and appropriate sounds.

```
1000 REM  **********************************
1010 REM *                                  *
1020 REM *          MUBBLE CHASE            *
1030 REM *                                  *
1040 REM *       GAMES ATARIS PLAY          *
1050 REM *                                  *
1060 REM *        COPYRIGHT 1983            *
1070 REM *         DATAMOST Inc.            *
1080 REM *                                  *
1090 REM  **********************************
```

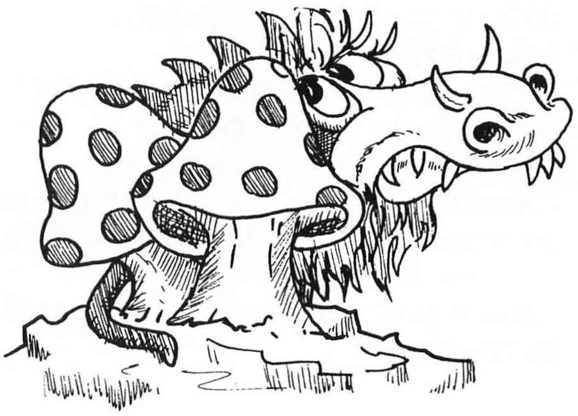


```
1100 REM
1110 REM DIMENSION STRINGS, VARIABLES
1120 DIM AN$(1),ME(3,3),SP(3),MU(3,3)
1130 GOSUB 1180:REM INSTRUCTIONS 1130
1140 GOSUB 1510:REM SETUP
1150 GOSUB 1810:REM PLAY
1160 GOSUB 2830:REM END
1170 END
1180 REM ****
1190 REM *** INSTRUCTIONS
1200 REM ****
1210 GRAPHICS 0
1220 POSITION 10,0:PRINT "*** MUBBLE CHASE ***"
1230 POSITION 2,5
1240 PRINT "IN THIS EXCITING GAME, YOU CONTROL"
1250 PRINT "THE MOVEMENT OF THE HUNGRY LITTLE"
1260 PRINT "CREATURE WE CALL THE MUBBLE."
1270 PRINT "THE MUBBLE SCURRIES THROUGH A MAZE,"
1280 PRINT "TRYING TO EAT ALL OF THE FOOD POINTS."
1290 PRINT
1300 PRINT "UNFORTUNATELY, THERE ARE THREE MUBBLE-"
1310 PRINT "EATERS IN THE SAME MAZE, WHO WANT TO"
1320 PRINT "CATCH AND EAT THE POOR MUBBLE."
1330 GOSUB 1490
1340 PRINT "{esc clear}":POSITION 10,0:PRINT
       "*** MUBBLE CHASE ***":POSITION 2,5
1350 PRINT "YOU MUST MANEUVER THE MUBBLE TO THE"
1360 PRINT "FOOD POINTS AND AWAY FROM THE MUBBLE"
1370 PRINT "EATERS. WHEN THESE NASTIES GET YOU,"
1380 PRINT "THE GAME IS OVER."
1390 PRINT
1400 PRINT "MOVEMENT OF THE MUBBLE IS CONTROLLED"
1410 PRINT "BY USING THE JOYSTICK"
1420 POSITION 19,14:PRINT "UP"
1430 POSITION 19,15:PRINT "{esc esc esc up arrow}"
1440 POSITION 14,16:PRINT "LEFT{esc esc esc left arrow}+{esc esc
       esc right arrow}RIGHT"
1450 POSITION 19,17:PRINT "{esc esc esc down arrow}"
1460 POSITION 18,18:PRINT "DOWN"
1470 GOSUB 1490
1480 RETURN
1490 POSITION 2,23:PRINT "PRESS RETURN TO CONTINUE ";:INPUT AN$
1500 RETURN
1510 REM ***
```

```
1520 REM ***SETUP
1530 REM ***
1540 GRAPHICS 5+16
1550 SETCOLOR 0,3,9
1560 COLOR 1
1570 PLOT 12,0
1580 DRAWTO 12,46
1590 DRAWTO 62,46
1600 DRAWTO 62,0
1610 DRAWTO 12,0
1620 SP(1)=2
1630 SP(2)=2
1640 SP(3)=2
1650 COLOR 3
1660 FOR I=15 TO 61 STEP 4
1670 PLOT I,3
1680 DRAWTO I,42
1690 NEXT I
1700 FOR I=3 TO 46 STEP 4
1710 PLOT 15,I
1720 DRAWTO 59,I
1730 NEXT I
1740 COLOR 2
1750 FOR I=3 TO 46 STEP 4
1760 FOR J=15 TO 59 STEP 4
1770 PLOT J,I
1780 NEXT J
1790 NEXT I
1800 RETURN
1810 REM ***
1820 REM ***PLAY
1830 REM ***
1840 MD=1
1850 ME(1,0)=3
1860 ME(1,1)=15
1870 ME(1,2)=3
1880 ME(2,0)=3
1890 ME(2,1)=59
1900 ME(2,2)=3
1910 ME(3,0)=3
1920 ME(3,1)=59
1930 ME(3,2)=43
1940 COLOR 1
1950 FOR I=1 TO 3
```



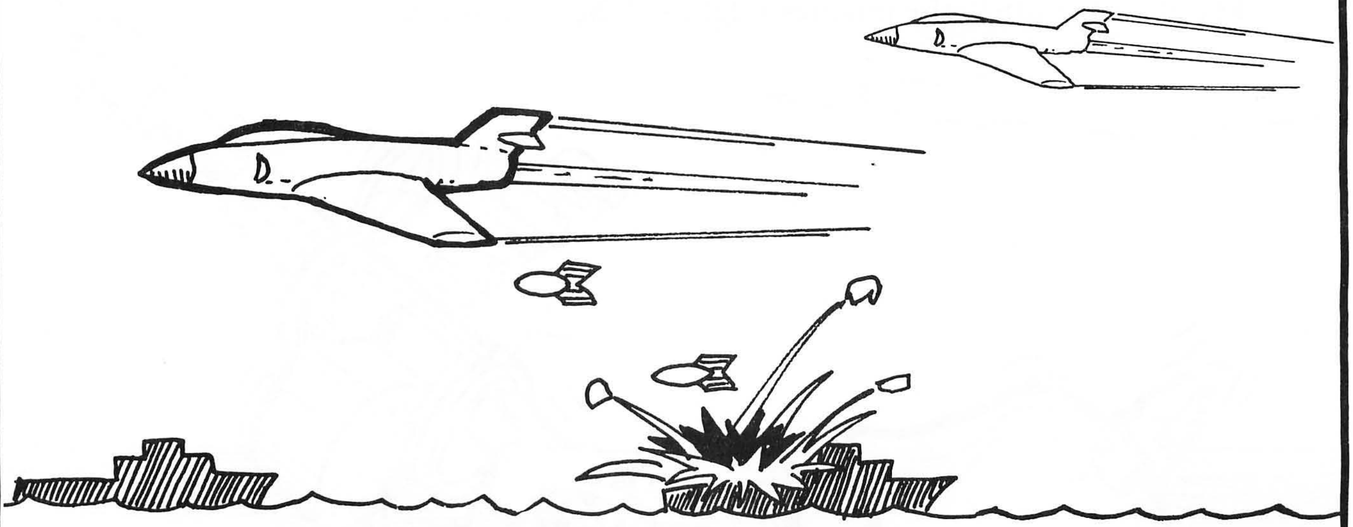
```
1960 PLOT ME(I,1),ME(I,2)
1970 NEXT I
1980 HI=0
1990 MU(1,1)=17
2000 MU(1,2)=43
2010 MU(2,1)=17
2020 MU(2,2)=43
2030 MU(3,1)=17
2040 MU(3,1)=17
2050 MU(3,2)=43
2060 MX=MU(1,1)
2070 MY=MU(1,2)
2080 FOR I=1 TO 10
2090 SOUND 0,123,10,10:COLOR 0
2100 PLOT MX,MY
2110 SOUND 0,211,10,10:COLOR 2
2120 PLOT MX,MY
2130 FOR PAUSE=1 TO 30:NEXT PAUSE
2140 NEXT I
2150 SOUND 0,0,0,0
2160 GOSUB 2210:REM MOVE MUBBLE
2170 IF PT=132 THEN 2830
2180 GOSUB 2550:REM MOVE MUBBLE EATER
2190 IF NOT HI THEN 2160
2200 GOTO 2830
2210 FL=((MX+1)/4)=INT(((MX+1)/4)) AND ((MY+1)/4)=INT(((MY+1)/4))
2220 IF NOT FL THEN 2300
2230 KEY=STICK(0)
2240 IF KEY=15 THEN 2300
2250 POKE 77,0:REM TURN OFF ATTRACT
2260 IF KEY=7 THEN MD=1
2270 IF KEY=14 THEN MD=2
2280 IF KEY=11 THEN MD=3
2290 IF KEY=13 THEN MD=4
2300 X2=MX:Y2=MY
2310 IF MD=1 THEN X2=X2+1
2320 IF MD=2 THEN Y2=Y2-1
2330 IF MD=3 THEN X2=X2-1
2340 IF MD=4 THEN Y2=Y2+1
2350 IF Y2>43 THEN Y2=43
2360 IF X2<15 THEN X2=15
2370 IF X2>59 THEN X2=59
2380 IF Y2<3 THEN Y2=3
2390 LOCATE X2,Y2,FD
```



```
2400 IF FD=2 THEN PT=PT+1:FOR I=250 TO 200 STEP -2:SOUND 0,I,10,10:NEXT I
2410 SOUND 0,0,0,0
2420 COLOR 3
2430 PLOT MU(1,1),MU(1,2)
2440 COLOR 1
2450 PLOT X2,Y2
2460 MU(1,1)=MU(2,1)
2470 MU(2,1)=MU(3,1)
2480 MU(1,2)=MU(2,2)
2490 MU(2,2)=MU(3,2)
2500 MU(3,1)=X2
2510 MU(3,2)=Y2
2520 MX=X2
2530 MY=Y2
2540 RETURN
2550 FOR I=1 TO 3
2560 A=ME(I,1):B=ME(I,2)
2570 FL=((A+1)/4)=INT(((A+1)/4)) AND ((B+1)/4)=INT(((B+1)/4))
2580 IF NOT FL THEN 2660
2590 ON INT(RND(1)*6)+1 GOTO 2600,2600,2600,2650,2660,2660
2600 ON INT(RND(1)*2)+1 GOTO 2610,2630
2610 IF MX>A THEN ME(I,0)=3:GOTO 2660
2620 IF MX>A THEN ME(I,0)=1:GOTO 2660
2630 IF MY<A THEN ME(I,0)=2:GOTO 2660
2640 IF MY<A THEN ME(I,0)=4:GOTO 2660
2650 ME(I,0)=INT(RND(1)*4)+1
2660 X2=A:Y2=B
2670 IF ME(I,0)=1 THEN X2=X2+1
2680 IF ME(I,0)=2 THEN Y2=Y2-1
2690 IF ME(I,0)=3 THEN X2=X2-1
2700 IF ME(I,0)=4 THEN Y2=Y2+1
2710 IF Y2>43 THEN Y2=43
2720 IF X2<15 THEN X2=15
2730 IF X2>59 THEN X2=59
2740 IF Y2<3 THEN Y2=3
2750 LOCATE A,B,KD:IF KD< >1 THEN SP(I)=KD
2760 COLOR SP(I):PLOT A,B
2770 LOCATE X2,Y2,Q:SP(I)=Q
2780 COLOR 1:PLOT X2,Y2:ME(I,1)=X2:ME(I,2)=Y2
2790 A=X2:B=Y2:C=MU(1,1):D=MU(1,2):E=MU(2,1):F=MU(2,2):G=MU(3,1)
     :H=MU(3,2)
2800 IF (A=C AND B=D) OR (A=E AND B=F) OR (A=G AND B=H) THEN HI=1
2810 NEXT I
```

```
2820 RETURN
2830 REM ***
2840 REM *** END GAME
2850 REM ***
2860 GRAPHICS 0
2870 POSITION 0,10
2880 IF PT<132 THEN GOTO 2900
2890 IF PT=132 THEN PRINT "CONGRATULATIONS PERFECT SCORE!!":RETURN
2900 PRINT "{esc clear}"
2910 GRAPHICS 17
2920 PRINT #6;"THOSE MUBBLE EATERS"
2930 PRINT #6;"GOT YOU THIS TIME":REM THIS LINE INVERSE
2940 PRINT #6;"BUT YOU DID EAT"
2950 FOR I=1 TO PT
2960 SOUND 0,I,10,10
2970 FOR PAUSE=1 TO 20:NEXT PAUSE
2980 PRINT #6;I;" ";
2990 NEXT I
3000 SOUND 0,0,0,0
3010 PRINT #6;"POINTS."
3020 FOR PAUSE=1 TO 1000:NEXT PAUSE
3030 END
```

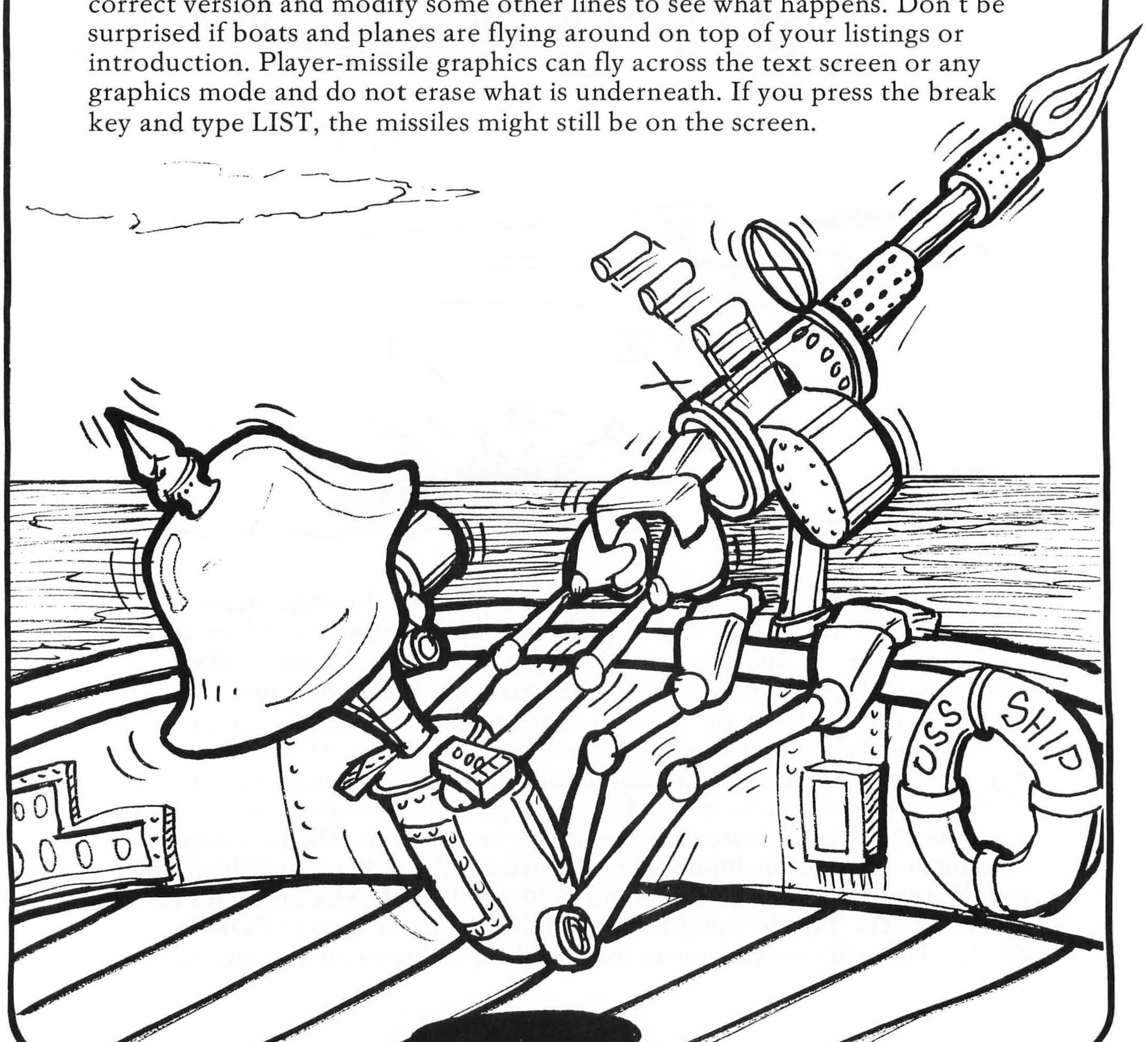
Air Attack



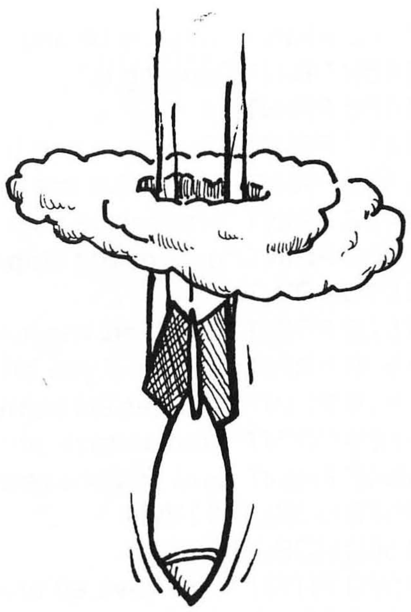
The Atari Home Computer gives you the capability of using machine language graphics in your BASIC programs. Player-missile graphics utilize special routines in Atari memory that are not available on most other computers. Player-missile graphics are extremely fast in machine language and reflect the ancestry of the Atari as an arcade machine. The graphics are slower when called up from BASIC, but are still relatively faster than straight BASIC, especially in the higher resolutions. If you are a beginning programmer, the best use of this game would be to change the data statements a few at a time until you have created a different airplane or ship. The first data statement, 2810, is the bomb, the next two are the airplane and the following are the explosion and the ship. Line 1940-2010 set the size and color of the player objects. Try changing 1980 to POKE S0,3 and 1990 to POKE S0+1,4 This will give you a very long airplane trying to hit a minuscule ship.

The terms player and missile refer to the size of the two kinds of objects that can be programmed in this mode, rather than any specific object. The missile can be only eight screen dots total, while the player can be 255 bytes of information arranged in a great variety of shapes. The sound routines are also interesting in this program because the screen position is used in the sound statement to make the boat and plane get louder when they are in the center of the screen. Change the value of DEMO to 1 in line 1140 and the airplane will never miss; kind of spooky when you think of the implications.

Save the original version of Air Attack on tape or disk, and if you mess up the version in the computer's memory no harm is done, just re-load the correct version and modify some other lines to see what happens. Don't be surprised if boats and planes are flying around on top of your listings or introduction. Player-missile graphics can fly across the text screen or any graphics mode and do not erase what is underneath. If you press the break key and type LIST, the missiles might still be on the screen.



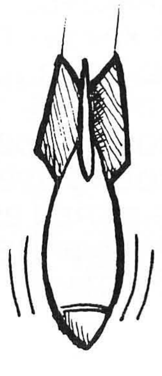
```
1000 REM  ***********************************
1010 REM *                                 *
1020 REM *           AIR ATTACK            *
1030 REM *                                 *
1040 REM  ***********************************
1050 REM
1060 REM THIS PROGRAM USES PLAYER
1070 REM MISSLE GRAPHICS
1080 REM
1090 REM
1100 REM IT ALSO USES SOUND
1110 REM
1120 REM DIMENSION VARIABLES
1130 DIM BM(4),SH(31)
1140 DEMO=0
1150 BM(1)=0
1160 GOTO 1390
1170 REM SKIP SUBROUTINES
1180 POSITION 2,23
1190 POKE 754,255
1200 PRINT "Press 'RETURN' when ready ";
1210 IF PEEK(754)< >12 THEN 1210
1220 POKE 754,255
1230 RETURN
1240 PRINT "}";
1250 POSITION 11,3
1260 PRINT "*** AIR ATTACK ***"
1270 POSITION 2,6
1280 RETURN
1290 FOR I=0 TO 3
1300 POKE MY+384+YB+I,BM(I)
1310 NEXT I
1320 RETURN
1330 B=0
1340 FOR I=0 TO 20
1350 POKE MY+384+80+I,0
1360 NEXT I
1370 YB=0
1380 RETURN
1390 GRAPHICS 0
1400 SL=15
1410 GOSUB 1240
1420 PRINT "In this game you are a fighter pilot."
1430 PRINT "You score by hitting one of the enemy"
```



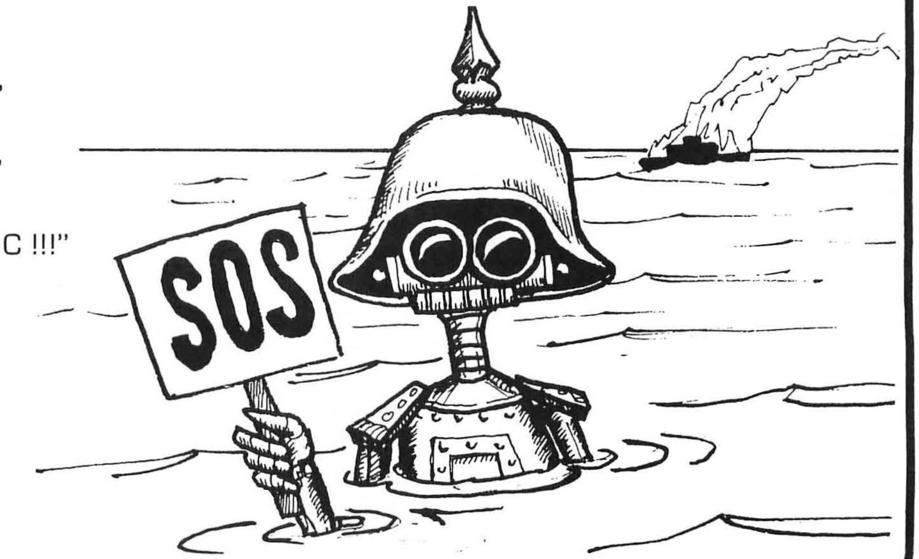
```
1440 PRINT "ships with one of your bombs and"
1450 PRINT "sinking it."
1460 PRINT
1470 PRINT "To drop a bomb simply press any key"
1480 PRINT "on the keyboard. Your score for"
1490 PRINT "hitting the ship will depend on which"
1500 PRINT "part of the ship you hit."
1510 PRINT
1520 PRINT "If you hit the lower deck you get 10"
1530 PRINT "points. If you hit the upper deck"
1540 PRINT "you get 20 points. If you hit the"
1550 PRINT "smokestack you have done very well,"
1560 PRINT "and you are rewarded with 30 points."
1570 GOSUB 1180
1580 GOSUB 1240
1590 PRINT "You have an arsenal of ";SL;" bombs."
1600 PRINT "The speed of each ship will vary so"
1610 PRINT "make every shot count!"
1620 PRINT
1630 PRINT "GOOD LUCK !!!"
1640 GOSUB 1180
1650 RESTORE
1660 FOR I=0 TO 3
1670 READ A:BM(I)=A
1680 NEXT I
1690 GRAPHICS 5
1700 COLOR 2
1710 SETCOLOR 2,10,5
1720 FOR Y=38 TO 39
1730 PLOT 0,Y
1740 DRAWTO 79,Y
1750 NEXT Y
1760 PM=54279:REM P-M BASE POINTER
1770 RA=106:REM OS TOP OF RAM
1780 SD=559:REM DMACTL SHADOW
1790 GR=53277:REM GTIA CONTROL
1800 H0=53248:REM H POSITION OF P0
1810 C0=704:REM COLOR OF P0
1820 S0=53256:REM SIZE OF P0
1830 A=PEEK(RA)-8:REM GET RAM 2K
1840 REM BELOW TOP
1850 POKE PM,A:REM TELL ANTIC WHERE
1860 REM PM RAM IS
1870 MY=256*A:REM PM RAM ADDRESS
```

```
1880 POKE SD,46:REM ENABLE DMA 2 LINE
1890 POKE GR,3:REM ENABLE PM DISPLAY
1900 FOR I=MY+384 TO MY+768
1910 POKE I,0
1920 NEXT I
1930 REM JUST CLEARED PLAYER RAM
1940 POKE C0,140
1950 POKE C0+1,104
1960 POKE H0,0
1970 POKE H1,0
1980 POKE S0,1
1990 POKE S0+1,3
2000 POKE S0+4,1
2010 FOR I=1 TO 6
2020 READ A:POKE MY+530+I,A
2030 NEXT I
2040 FOR I=1 TO 6
2050 READ A
2060 POKE MY+725+I,A
2070 SH(I)=A
2080 NEXT I
2090 FOR I=7 TO 31
2100 READ A
2110 SH(I)=A
2120 NEXT I
2130 SC=0
2140 X1=0
2150 X2=0
2160 S2=3
2170 PRINT "}SCORE          ";SC
2180 PRINT "SHOTS LEFT ";SL
2190 IF SL=0 THEN 2890
2200 POKE H0,X1
2210 POKE H0+1,X2
2220 X1=X1-1.5
2230 IF X1<40 THEN X1=190
2240 X2=X2+S2
2250 IF X2<191 THEN 2280
2260 S2=INT(RND(1)*3)+1
2270 X2=40
2280 SOUND 1,50-S2,4,3-ABS((X2-127)/128)*3
2290 SOUND 0,20,8,3-ABS((X1-127)/128)*3
2300 IF B THEN 2560
2310 IF S2>0 THEN 2450
2320 CT=CT+1
```

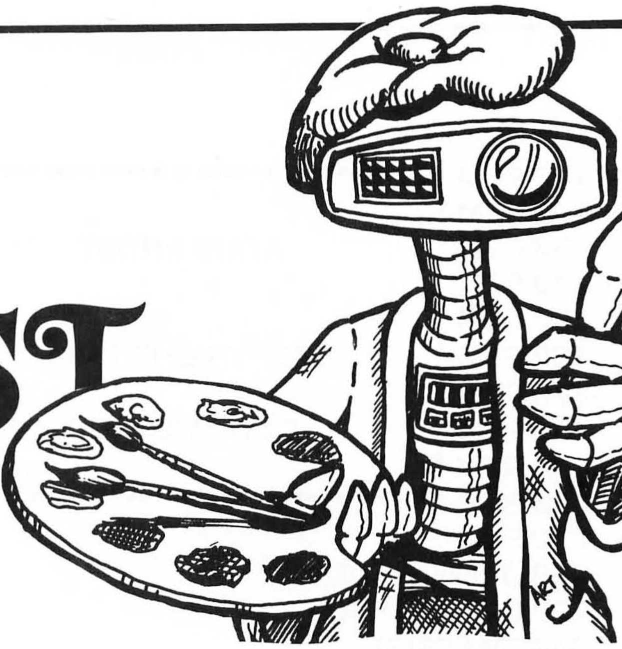
```
2330 FOR I=1 TO 6
2340 POKE MY+725+I,SH(CT*6+I)
2350 NEXT I
2360 SOUND 2,CT*15+RND(1)*190,4,8
2370 IF CT=4 THEN CT=-1:GOTO 2420
2380 IF CT>0 THEN 2450
2390 S2=3
2400 X2=255
2410 SOUND 2,0,0,0
2420 X2=1
2430 POKE H0+1,0
2440 REM
2450 IF NOT DEMO THEN 2490
2460 IF X1-30<X2+S2*35 THEN 2490
2470 IF X1-30>X2+S2*35+4 THEN 2490
2480 GOTO 2510
2490 IF PEEK(754)=255 THEN 2200
2500 POKE 754,255
2510 B=1:YB=20
2520 SL=SL-1
2530 POKE 53278,0:REM CLEAR COLLISIONS
2540 X3=X1+3
2550 POKE H0+4,X3
2560 IF YB<91 AND S2 THEN 2610
2570 GOSUB 1330
2580 POKE 754,255
2590 SOUND 2,0,0,0
2600 GOTO 2170
2610 POKE 540,2
2620 IF PEEK(540) THEN 2620
2630 CL=PEEK(S0)
2640 POKE 53278,0:REM CLEAR COLLISIONS
2650 IF CL< >2 THEN 2740
2660 IF YB=86 THEN P=30
2670 IF YB=88 THEN P=20
2680 IF YB=90 THEN P=10
2690 S2=0
2700 SC=SC+P
2710 CT=0
2720 GOSUB 1330
2730 GOTO 2170
2740 SOUND 2,20+YB,10,4
2750 GOSUB 1290
2760 POKE H0+4,X3
```



```
2770 X3=X3-0.75
2780 IF X3<40 THEN X3=190
2790 YB=YB+2
2800 GOTO 2200
2810 DATA 0,0,1,1
2820 DATA 1,1,3,63,115,127
2830 DATA 4,4,14,14,63,63
2840 DATA 0,18,33,18,51,12
2850 DATA 0,0,33,18,63,12
2860 DATA 0,0,12,12,30,12
2870 DATA 0,0,0,0,0,18
2880 DATA 0,0,0,0,0,0
2890 GRAPHICS 0:GOSUB 1240
2900 PRINT "GAME OVER !!!"
2910 SOUND 0,0,0,0
2920 SOUND 1,0,0,0
2930 SOUND 2,0,0,0
2940 PRINT "YOUR SCORE OF ";SC;" IS ";
2950 POKE 559,34
2960 POKE 53277,0
2970 FOR I=53261 TO 53265
2980 POKE I,0
2990 NEXT I
3000 ON SC/25 GOTO 3010,3040,3060,3080,3100,3120,3140
3010 IF SC/25>6 THEN 3140
3020 PRINT "ROTTEN !!!"
3030 END
3040 PRINT "BAD !!!"
3050 END
3060 PRINT "POOR !!!"
3070 END
3080 PRINT "FAIR . . . "
3090 END
3100 PRINT "GOOD . . . "
3110 END
3120 PRINT "GREAT !!!"
3130 END
3140 PRINT "FANTASTIC !!!"
3150 END
```



ATARI ARTIST



Here's a simple program that will allow you to rival Picasso without getting paint all over the house. The video monitor is your canvas, the keyboard your palette, and the joystick and/or keyboard your brush. The program can save your drawing on disk or tape and load it back in to work on some more or to use as a background for a game or a title screen. When the picture is loaded you can select a different background color.

A single long play loop of IF-THEN statements looks for the input. The paddle is checked first. If it returns a value of 15 it is in the neutral position (not being moved) and the program jumps to 3095 to read the keyboard. If a paddle input is detected the program assigns the variable X to the equivalent arrow keys. Then the program jumps around the command to get keyboard input and carries out the command assigned to X. If the joystick isn't used, line 3095 GETs the keyboard input from #1 channel and assigns X to the ASCII value of the key. The keyboard arrows are actually on the keys that are assigned to "– " "= " "+ " and "* " and these are the characters that are looked for.

In 3400-3450 a color is set, a point plotted and a note sounded. Line 3600 empties out the X variable and 3650 sends the program back around the loop.

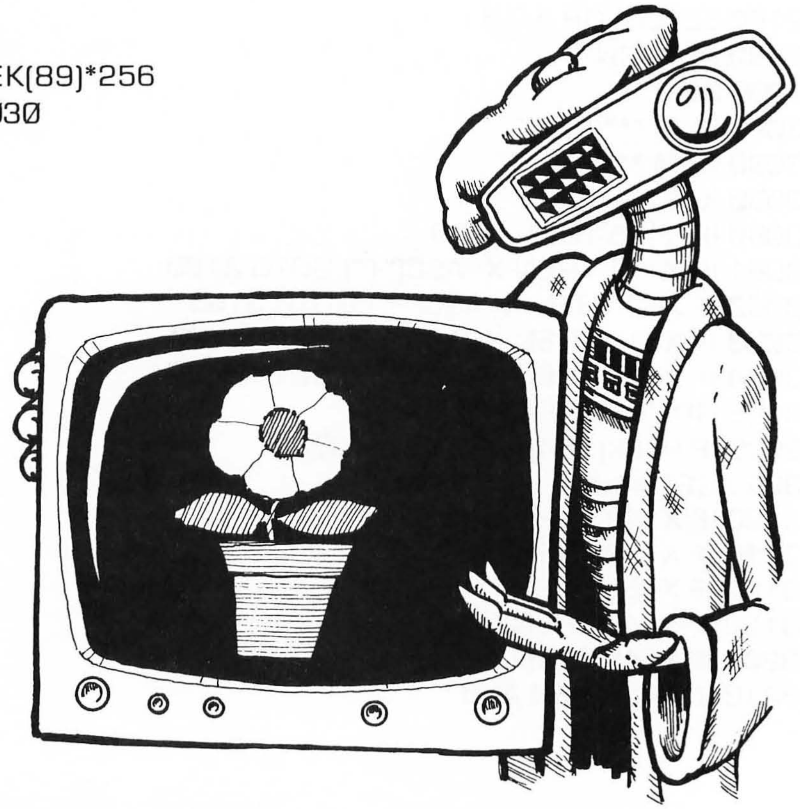
Lines 4200 to 4270 are a routine to save the picture dot by dot onto disk or tape. You can use this routine in any situation where you want to save a screen image, even text and graphics mixtures.

If you use an Atari Artist drawing in another program, be sure to use the GRAPHICS 7 mode or the drawing will come back as the most colorful garbage you have ever seen. In your own programs match the graphic modes of the saving and retrieving routines.

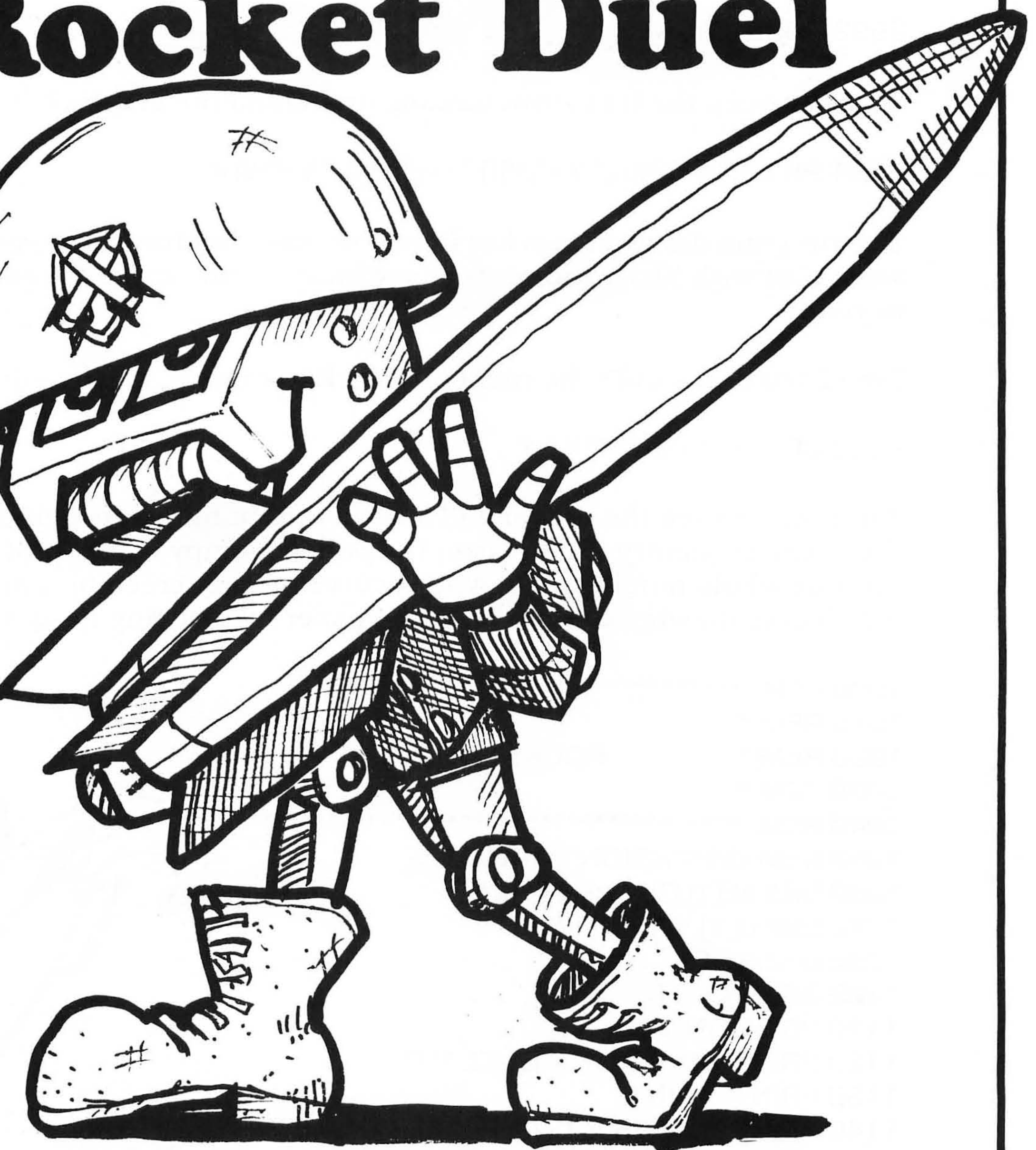
```
100 REM
110 REM ************************************
120 REM *                                  *
130 REM *           ATARI ARTIST           *
140 REM *                                  *
150 REM *                                  *
160 REM *          COPYRIGHT 1983          *
170 REM *                                  *
180 REM *          DATAMOST INC.           *
190 REM *                                  *
200 REM ************************************
210 REM
220 OPEN #1,4,0,"K:"
230 C=2:A=0:B=0
240 DIM C$(3)
250 DIM AN$(4),FI$(40)
260 A=PEEK(88)+PEEK(89)*256
270 TRAP 2100
300 GOSUB 1000:REM INTRODUCTION
310 GOSUB 2000:REM SETUP
320 GOSUB 3000:REM PLAY
330 GOSUB 4000:REM END ROUTINE
340 END
1000 REM ***
1010 REM *** INSTRUCTIONS ***
1020 REM ***
1030 GRAPHICS 0:PRINT "}"
1040 PRINT :PRINT
1050 PRINT "          *** ATARI ARTIST ***"
1060 PRINT :PRINT
1100 PRINT "THE COMMANDS ARE:"
1110 PRINT :PRINT
1130 PRINT "B=Select background"
1140 PRINT "S=Turn sound off/on"
1150 PRINT "F=Fill"
1160 PRINT "D=Done,save picture"
1170 PRINT :PRINT
1180 PRINT "Arrow keys move line"
1190 PRINT "0=Background color line"
1200 PRINT "1=Orange line"
1210 PRINT "2=Green line"
1230 PRINT "3=Blue line"
1240 PRINT :PRINT
1250 PRINT "Would you like to load a picture";
```

```
1260 INPUT AN$
1270 IF AN$(1,1)="Y" THEN GOTO 1300
1280 RETURN
1300 PRINT "Enter C:OR D: plus the picture name."
1310 INPUT FI$
1315 GRAPHICS 7
1320 OPEN #2,4,0,FI$
1325 A=PEEK(88)+PEEK(89)*256
1330 FOR I=A TO A+4030
1340 GET #2,X
1350 POKE I,X
1360 NEXT I
1370 CLOSE #2
1990 POP
1995 GOTO 2100
2000 REM ***
2010 REM *** SETUP
2020 REM ***
2030 GRAPHICS 7
2100 PRINT "ENTER BACKGROUND COLOR"
2110 PRINT "0=BLK 1=GRN 2=BRN 3&4=RED 5=MAGENTA"
2120 PRINT "6=PRPL 7,8&9=BLUE 10=GRAY 11=GRN"
2140 INPUT C$
2150 C=VAL(C$)
2160 SETCOLOR 4,C,0
2990 RETURN
3000 REM ***
3020 REM *** PLAY
3030 REM ***
3050 X=STICK(0)
3060 IF X=15 THEN 3090
3061 IF X=11 THEN X=ASC("+"):GOTO 3100
3062 IF X=7 THEN X=ASC("*"):GOTO 3100
3063 IF X=14 THEN X=ASC("-"):GOTO 3100
3064 IF X=13 THEN X=ASC("="):GOTO 3100
3065 GOTO 3050
3090 IF PEEK(764)=255 THEN 3050
3095 GET #1,X
3100 IF X=49 THEN C=1
3110 IF X=50 THEN C=2
3120 IF X=51 THEN C=3
3130 IF X=48 THEN C=4
3200 IF X=45 THEN A=A-1
3210 IF A<0 THEN A=0
```

```
3230 IF X=61 THEN A=A+1
3240 IF A>79 THEN A=79
3300 IF X=43 THEN B=B-1
3310 IF B<0 THEN B=0
3320 IF X=42 THEN B=B+1
3330 IF B>158 THEN B=158
3340 IF X=66 THEN GOSUB 2100
3350 IF X=83 THEN S= NOT S
3360 IF X=70 THEN POKE 765,C:XIO 18,#6,0,0,"S:"
3370 IF X=68 THEN RETURN
3400 COLOR C
3410 PLOT B,A
3430 IF S THEN SOUND 0,0,0,0:SOUND 1,0,0,0:GOTO 3600
3440 SOUND 0,B,10,10
3450 SOUND 1,A,10,10
3600 X=0
3650 GOTO 3020
4000 REM ***
4010 REM *** END ROUTINE ***
4020 REM ***
4040 PRINT "Do you want to save this picture";
4050 INPUT AN$
4060 IF AN$(1,1)< >"Y" THEN RETURN
4100 PRINT "Put C: OR D: in front of the picture name"
4120 INPUT FI$
4200 OPEN #2,8,0,FI$
4210 A=PEEK(88)+PEEK(89)*256
4240 FOR I=A TO A+4030
4250 PUT #2,PEEK(I)
4260 NEXT I
4270 CLOSE #2
```



Rocket Duel



The earliest electronic computers were put to work plotting trajectories for cannons and anti-aircraft guns. Your ATARI is as powerful as a machine that once took up an entire building and used as much electricity as a small city. The ROCKET DUEL game plots trajectories with SINe and COSine functions that are built into the BASIC language. You could draw your own landscape using data statements, and the scale of the rockets would be appropriate. You can see the rapid calculations the computer is doing by printing out the value in the middle of the rocket loop at 2690. Add a position and print statement:

```
2692 POSITION 23,1
```

This will keep the text from scolling up behind the graphics area.

```
2694 PRINT "X=";X(I);" Y=";Y(I);" U=";U(I);" V=";V(I)
```

The program draws the rocket in yellow, then 'undraws' the rocket in the same spot with black and plots a new location one space over to be redrawn in yellow.

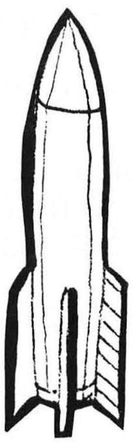
Line 2880 draws over the rocket in black to erase it. Change this line to:

```
288Ø COLOR 4:PLOT OX,OY
```

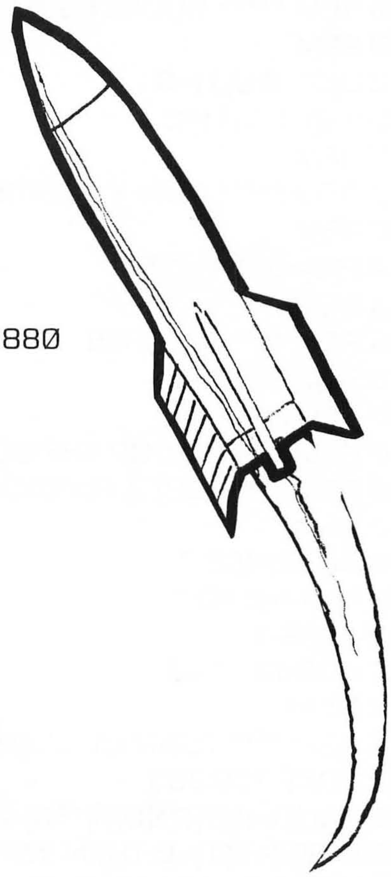
After this change the rocket will leave a trail of blue dots behind that show the entire trajectory. The reason the path is bumpy is that plot locations must be whole numbers. Drawing a curve on the screen presents the same problem as drawing a curve on graph paper by blacking in squares.

```
1ØØØ REM  ***********************************
1Ø1Ø REM *                                  *
1Ø2Ø REM *            ROCKET DUEL           *
1Ø3Ø REM *                                  *
1Ø4Ø REM  ***********************************
1Ø5Ø REM DIMENSION VARIABLES
1Ø6Ø DIM EL(1),OL(1),X(1),Y(1)
1Ø7Ø DIM U(1),V(1),P(1),S(1)
1Ø8Ø DIM A$(5)
11ØØ GRAPHICS Ø
111Ø POSITION 1Ø,3
112Ø PRINT "*** ROCKET DUEL ***"
113Ø PRINT :PRINT
114Ø PRINT "THIS IS A TWO PLAYER GAME WHERE YOU"
115Ø PRINT "USE YOUR JOYSTICKS TO AIM AND FIRE"
116Ø PRINT "ROCKETS AT THE OTHER PLAYERS LAUNCHER."
117Ø PRINT "THE SCREEN DISPLAYS WIND VELOCITY AND"
118Ø PRINT "LAUNCHER ANGLE."
119Ø PRINT
12ØØ POSITION 2,23
121Ø PRINT "PRESS 'RETURN' TO CONTINUE.";
122Ø INPUT A$
2Ø1Ø REM
2Ø5Ø REM
2Ø6Ø REM ***** STARTING ANGLE
```

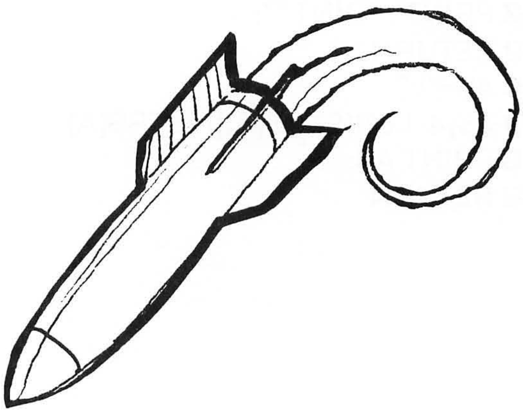
```
2070 REM
2080 DEG
2090 EL(0)=20:EL(1)=20
2100 OL(0)=0:OL(1)=0
2110 REM
2120 REM ***** ROCKET START POSITION
2130 REM
2140 X(0)=0:X(1)=0
2150 Y(0)=0:Y(1)=0
2160 REM
2170 REM ***** ROCKET SPEED
2180 REM
2190 U(0)=0:U(1)=0
2200 V(0)=0:V(1)=0
2210 REM
2220 REM ***** GUN X POSITIONS
2230 REM
2240 P(0)=5:P(1)=154
2250 REM
2260 REM ***** SCORES
2270 S(0)=0:S(1)=0
2280 REM
2290 REM ***** SETUP THE SCREEN AND
2300 REM ***** GET A RANDOM WIND SPEED
2310 REM
2320 GRAPHICS 7
2330 FOR I=0 TO 1
2340 J=I*2-1
2350 GOSUB 3140
2360 NEXT I
2370 REM **** TURN OF CURSOR
2380 POKE 752,255
2390 WIND=INT(RND(0)*50)-25
2400 IF S(0)+S(1)=0 THEN WIND=0
2410 PRINT "}";:GOSUB 3400
2420 REM
2430 REM **** TOP OF MAIN LOOP
2440 REM
2450 K=0
2460 FOR I=0 TO 1
2470 J=I*2-1
2480 REM
2490 REM READ STICK UP/DOWN
2500 REM
```



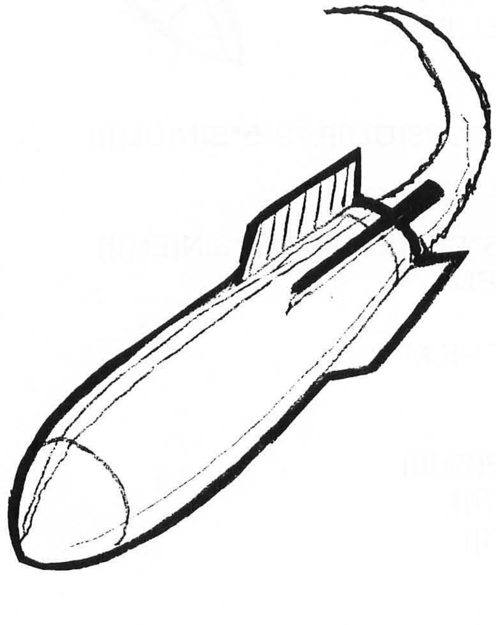
```
2510 JS=STICK(I)
2520 IF JS=15 THEN 2600
2530 IF JS=13 THEN EL(I)=EL(I)-1
2540 IF JS=14 THEN EL(I)=EL(I)+1
2550 IF JS=13 OR JS=14 THEN GOSUB 3140
2560 POKE 77,0
2570 REM
2580 REM ***** READ FIRE BUTTON
2590 REM
2600 IF X(I)+STRIG(I)=0 THEN GOSUB 3270
2610 REM
2620 REM ***** MOVE ROCKET
2630 REM
2640 IF X(I)=0 THEN 2880
2650 IF Y(I)>=0 THEN OX=X(I):OY=Y(I)
2660 X(I)=X(I)-U(I)
2670 U(I)=U(I)-WIND/10000
2680 Y(I)=Y(I)-V(I)
2690 V(I)=V(I)-0.02
2700 IF (X(I)<1)+(X(I)>159) THEN X(I)=0:GOTO 2880
2710 IF Y(I)>=0 THEN COLOR 1:PLOT X(I),Y(I)
2720 IF Y(I)<79 THEN 2880
2730 REM
2740 REM ***** ROCKET HIT LAUNCHER ?
2750 REM
2760 FOR K=15 TO 0 STEP -0.2
2770 SOUND 0,20,8,K
2780 SOUND 1,200,8,K
2790 NEXT K
2800 K=0
2810 IF ABS(P(1-I)-X(I))<3 THEN S(I)=S(I)+1:K=1
2820 IF ABS(P(I)-X(I))<3 THEN S(I)=S(I)-1:K=1
2830 X(I)=0
2840 IF K=1 THEN I=1
2850 REM
2860 REM ***** BOTTOM OF LOOP
2870 REM
2880 COLOR 0:PLOT OX,OY
2890 IF X(I)>0 THEN 2910
2900 SOUND 2+I,0,0,0
2910 NEXT I
2920 IF K=1 THEN 2320
2930 REM
2940 REM ***** CHECK FOR ROCKETS HIT
```

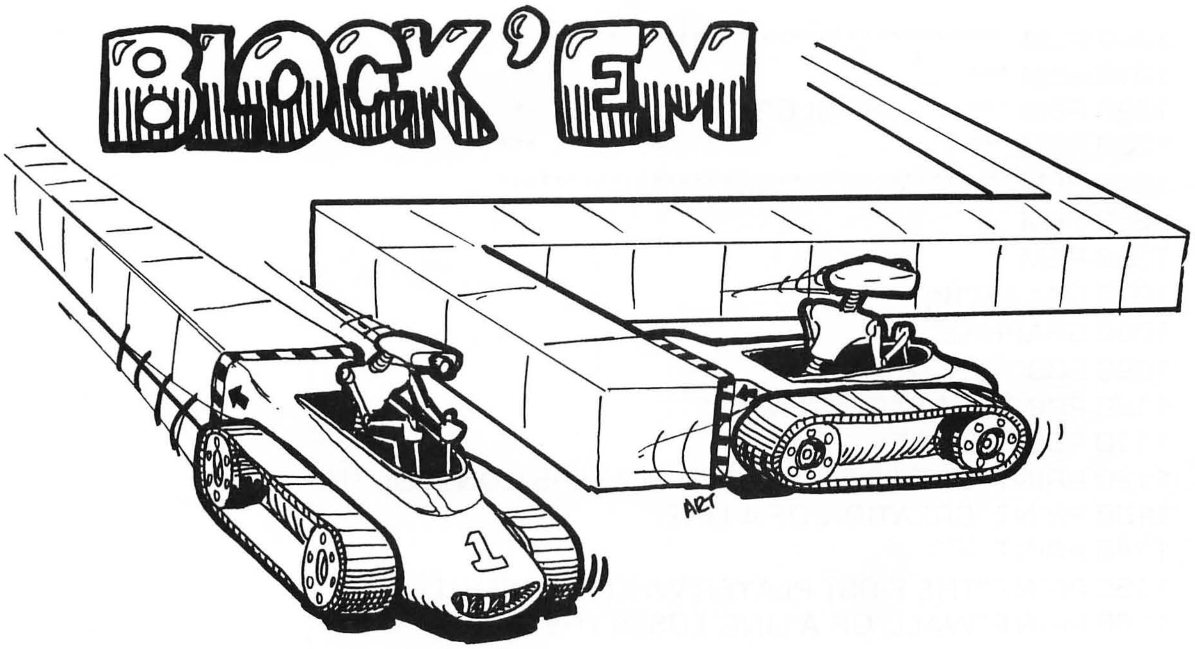


```
2950 IF INT(Y(0)/4)< >INT(Y(1)/4) THEN 3090
2960 IF INT(X(0)/4)< >INT(X(1)/4) THEN 3090
2970 IF X(0)=0 OR X(1)=0 THEN 3090
2980 SOUND 1,90,8,10
2990 SOUND 2,0,0,0
3000 COLOR 0
3010 PLOT X(0),Y(0)
3020 PLOT X(1),Y(1)
3030 X(0)=0:X(1)=0
3040 Y(0)=0:Y(1)=0
3050 FOR X=9 TO 0 STEP -1
3060 FOR Y=1 TO 10:NEXT Y
3070 SOUND 1,90,8,X
3080 NEXT X
3090 GOTO 2450
3100 REM
3110 REM ***** CLEAR AND REDRAW
3120 REM ***** ROCKET LAUNCHER
3130 REM
3140 GOSUB 3400
3150 IF EL(I)>90 THEN EL(I)=90
3160 IF EL(I)<0 THEN EL(I)=0
3170 COLOR 0
3180 PLOT P(I),79
3190 DRAWTO P(I)-J*5*COS(OL(I)),79-5*SIN(OL(I))
3200 COLOR 1
3210 PLOT P(I),79
3220 DRAWTO P(I)-J*5*COS(EL(I)),79-5*SIN(EL(I))
3230 OL(I)=EL(I):RETURN
3240 REM
3250 REM ***** PLOT THE MISSLE
3260 REM
3270 POKE 77,0
3280 X(I)=P(I)-J*5*COS(EL(I))
3290 Y(I)=79-5*SIN(EL(I))
3300 U(I)=J*2*COS(EL(I))
3310 V(I)=2*SIN(EL(I))
3320 SOUND 0,45,8,7
3330 SOUND 1,50,8,7
3340 FOR K=7 TO 0 STEP -1
3350 FOR Y=1 TO 15:NEXT Y
3360 SOUND 0,45,8,K
3370 SOUND 1,50,8,K
3380 NEXT K
```



```
339Ø SOUND 2+I,1Ø+3Ø*RND(1),8,5
34ØØ A=EL(Ø):GOSUB 353Ø
341Ø PRINT " ";
342Ø A=S(Ø):GOSUB 353Ø
343Ø PRINT "          WIND ";
344Ø A=WIND:GOSUB 353Ø
345Ø IF WIND>Ø THEN PRINT "-";
346Ø IF WIND<=Ø THEN PRINT " ";
347Ø PRINT "          ";
348Ø A=EL(1):GOSUB 353Ø
349Ø PRINT " ";
35ØØ A=S(1):GOSUB 353Ø
351Ø PRINT :PRINT "";
352Ø RETURN
353Ø A$="   "
354Ø A$(4-LEN(STR$(A)))=STR$(A)
355Ø PRINT A$;
356Ø RETURN
```



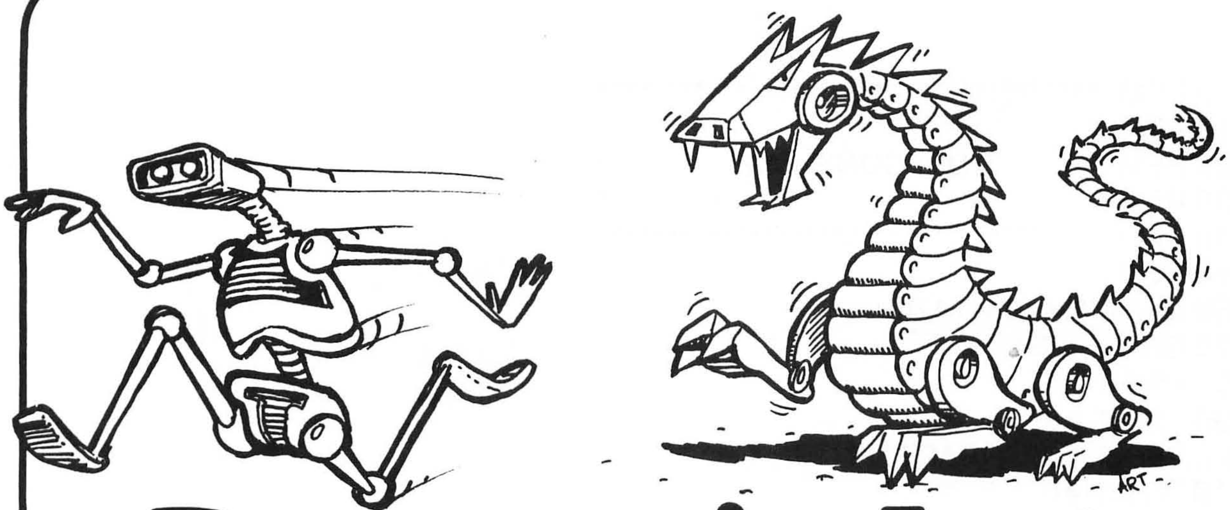


This is a two player game, so get ready for some stiff competition! You will be surprised at how much strategy it takes to outwit a determined opponent. The computer will play against you, but the computer advisary lacks the killer instinct and is easy to beat. The main program loop starts at 1470. The delay variable [DL] is reduced by one at each pass so that play speeds up as the game progresses. There is a single routine that reads the joystick and tests the output to see if it is within bounds or not crashing into the opponent's line. The program demonstrates the use of conditional branching. You are asked to choose one or two players, and your response is stored as DEMO=1 [true or yes] or DEMO=0 [false or no]. Later the program uses 1590 IF NOT DEMO THEN 1760 to branch around the computer play section of the program. The IF THEN statement can be very useful. Look at line 1260 IF LEN (A$) THEN A=VAL(A$). If you press RETURN without choosing 1 or 2, A$ will be empty and LEN will be 0. The action that follows THEN [A=VAL(A$)] will not be carried out, and line 1280 will send the program around again until an acceptable answer is received. These are both instances of conditional branching. If you are getting into the spirit of programming, you could add a few lines to the computer play section to make the computer more competitive.

```
1000 REM  ************************************
1010 REM ***                              ***
1020 REM ***           BLOCK 'EM          ***
1030 REM ***                              ***
1040 REM  ************************************
1050 REM
1060 REM
1070 DIM A$(10)
1080 GRAPHICS 0
1090 POSITION 12,2
1100 PRINT "*** BLOCK 'EM ***"
1110 POSITION 2,5
1120 PRINT "IN THIS GAME, TWO PLAYERS CONTROL THE"
1130 PRINT "CREATION OF A LINE."
1140 PRINT
1150 PRINT "THE FIRST PLAYER WHOSE LINE HITS A"
1160 PRINT "WALL, OR A LINE, LOSES THE GAME."
1170 PRINT
1180 PRINT "USE THE JOYSTICKS TO CONTROL THE GAME"
1190 PRINT "FOR UP,DOWN,LEFT AND RIGHT."
1200 PRINT
1210 PRINT "          GOOD LUCK !!"
1220 PRINT
1230 DEMO=0
1240 PRINT "HOW MANY PLAYERS 1,2 ";
1250 INPUT A$
1260 IF LEN(A$) THEN A=VAL(A$)
1270 IF A=2 THEN 1300
1280 IF A< >1 THEN 1220
1290 DEMO=1
1300 POSITION 2,23
1310 PRINT "PRESS 'RETURN' WHEN READY. ";
1320 INPUT A$
1330 GRAPHICS 5
1340 COLOR 2
1350 PLOT 2,0
1360 DRAWTO 79,0
1370 DRAWTO 79,39
1380 DRAWTO 2,39
1390 DRAWTO 2,0
1400 X1=20
1410 X2=60
1420 A1=7
1430 Y1=20
```

```
1440 Y2=20
1450 A2=11
1460 DL=100
1470 FOR TM=1 TO DL
1480 NEXT TM
1490 DL=DL-1
1500 IF DL<1 THEN DL=1
1510 COLOR 1
1520 PLOT X1,Y1
1530 COLOR 3
1540 PLOT X2,Y2
1550 D1=STICK(0)
1560 CH=0
1570 RN=0
1580 D2=A2
1590 IF NOT DEMO THEN 1760
1600 YC=Y2:IF A2=14 THEN YC=Y2-1
1610 IF A2=13 THEN YC=Y2+1
1620 XC=X2:IF A2=11 THEN XC=X2-1
1630 IF A2=7 THEN XC=X2+1
1640 LOCATE XC,YC,KD
1650 IF KD=0 AND (DL=1 OR RN) THEN 1770
1660 IF CH=2 THEN A2=D2:GOTO 1770
1670 IF KD THEN GOSUB 2140:GOTO 1600
1680 D3=INT(RND(1)*15)+1
1690 RN=1
1700 IF D3>14 THEN 1770
1710 IF D3<7 THEN 1770
1720 IF D3>7 AND D3<11 THEN 1770
1730 IF D3=12 THEN 1770
1740 A2=D3
1750 GOTO 1600
1760 D2=STICK(1)
1770 SOUND 0,50,10,5
1780 IF DEMO THEN D2=15
1790 IF D1=15 THEN 1840
1800 IF D1<7 THEN 1840
1810 IF D1=9 THEN 1840
1820 IF D1=10 THEN 1840
1830 A1=D1
1840 IF D2=15 THEN 1890
1850 IF D2<7 THEN 1890
1860 IF D2=9 THEN 1890
1870 IF D2=10 THEN 1890
1880 A2=D2
```

```
1890 IF A1=14 THEN Y1=Y1-1
1900 IF A2=14 THEN Y2=Y2-1
1910 IF A1=13 THEN Y1=Y1+1
1920 IF A2=13 THEN Y2=Y2+1
1930 IF A1=11 THEN X1=X1-1
1940 IF A2=11 THEN X2=X2-1
1950 IF A1=7 THEN X1=X1+1
1960 IF A2=7 THEN X2=X2+1
1970 LOCATE X1,Y1,C1
1980 LOCATE X2,Y2,C2
1990 IF NOT C1 AND NOT C2 THEN 2090
2000 SOUND 0,0,0,0
2010 IF C1 THEN PRINT "PLAYER ONE LOSES"
2020 IF C2 THEN PRINT "PLAYER TWO LOSES"
2030 IF NOT C1 OR NOT C2 THEN 2050
2040 PRINT "YOU BOTH LOST"
2050 PRINT "PLAY AGAIN Y/N ";
2060 INPUT A$
2070 IF LEN(A$) THEN IF A$(1,1)="Y" THEN 1330
2080 END
2090 COLOR 2
2100 PLOT X1,Y1
2110 PLOT X2,Y2
2120 SOUND 0,0,0,0
2130 GOTO 1470
2140 CH=CH+1
2150 RN=1
2160 ON CH GOTO 2200,2290
2170 D=INT(RND(1)*999)
2180 D=D/2-INT(D/2)
2190 D=D*2
2200 ON D GOTO 2210,2250
2210 IF A2=14 THEN A2=11:RETURN
2220 IF A2=7 THEN A2=14:RETURN
2230 IF A2=13 THEN A2=7:RETURN
2240 IF A2=11 THEN A2=13:RETURN
2250 IF A2=14 THEN A2=13:RETURN
2260 IF A2=7 THEN A2=11:RETURN
2270 IF A2=13 THEN A2=14:RETURN
2280 IF A2=11 THEN A2=7:RETURN
2290 IF A2=14 THEN A2=13:RETURN
2300 IF A2=11 THEN A2=7:RETURN
2310 IF A2=13 THEN A2=14:RETURN
2320 IF A2=7 THEN A2=11:RETURN
```



Dragon's Lair

Maze games of all kinds have been a mainstay of microcomputers from the very beginning, with SNACK-ATTACK™ and PAC-MAN™ as the descendants of a large family. Dragon's Lair is the grandfather, and our version of it is sort of a great-uncle. The special feature of our game is that after the maze is drawn the lights go out in the lair. You have to memorize the maze as it appears. When you bump into a wall that section of the wall becomes visible, but you won't have enough time to get through the maze by hit or miss. The dragon starts from the exit you are trying to reach and is drawn inexorably toward you with super radar. Eluding the dragon is more a matter of tactics than dexterity. If you backtrack around a corner and get a wall between you and the dragon you could end up with Mr. Dragon behind you and the exit will then be in the clear. Dragon's Lair has a two player option that allows the second player to become the dragon and attack player one and/or head for the opposite door.

The maze is built with graphic characters in the text mode, and runs relatively fast for a BASIC program. The program POKEs in the characters for the maze. In line 1250, POKE X+Y,128 places Atari character 128 on the screen, which is a white square. Try changing 128 to 160 and the maze will be drawn with inverse colons. Change 120 to 84 to draw white circles. A grid is drawn first, then series of random ON GOTOs in line 1610 sends the program to lines that erase the walls in the grid to form a maze. Try changing the 0 in line 1660 to some other value and watch the screen draw in characters instead of erasing parts of the grid. The keyboard and joystick reading sections can be utilized in any of your own programs that need to move a figure on the screen. You can change or add to the sounds, change colors, and make a different dragon if you want a personal version of Dragon's Lair.

```
1000 REM ************************************
1010 REM *                                  *
1020 REM *          DRAGON'S LAIR           *
1030 REM *                                  *
1040 REM ************************************
1050 REM
1060 DIM M(14,8),T(14,8)
1070 GRAPHICS 2
1080 PRINT #6
1090 PRINT #6
1100 PRINT #6;"  DRAGONS LAIR"
1110 PRINT #6
1120 PRINT #6;" 1=1 PLAYER KB"
1130 PRINT #6;" 2=1 PLAYER STICK"
1140 PRINT #6;" 3=2 PLAYER KB"
1150 PRINT #6;" 4=2 PLAYER STICK"
1160 INPUT OPT
1170 POKE 40000,0
1180 SCREEN=40080
1190 REM BUILD MAZE
1200 POKE 82,0
1210 GRAPHICS 0
1220 POKE 752,1
1230 FOR X=SCREEN TO SCREEN+840 STEP 120
1240 FOR Y=0 TO 39
1250 POKE X+Y,128
1260 NEXT Y
1270 NEXT X
1280 FOR X=SCREEN TO SCREEN+39 STEP 3
1290 FOR Y=0 TO 840 STEP 40
1300 POKE X+Y,128
1310 NEXT Y
1320 NEXT X
1330 S=150
1340 FOR I=0 TO 14
1350 FOR J=0 TO 8
1360 T(I,J)=0
1370 M(I,J)=11
1380 NEXT J
1390 NEXT I
1400 X=INT(RND(0)*13+1)
1410 Y=INT(RND(0)*7+1)
1420 C=91
```

```
1430 IF C=1 THEN 1980
1440 R=0
1450 D=0
1460 L=0
1470 U=0
1480 M(X,Y)=-ABS(M(X,Y))
1490 C=C-1
1500 IF X=13 THEN 1520
1510 IF M(X+1,Y)>0 THEN R=1
1520 IF Y=7 THEN 1540
1530 IF M(X,Y+1)>0 THEN D=1
1540 IF X=1 THEN 1560
1550 IF M(X-1,Y)>0 THEN L=1
1560 IF Y=1 THEN 1580
1570 IF M(X,Y-1)>0 THEN U=1
1580 Q=R+D+L+U
1590 IF (Q<3 AND INT(RND(0)*10)<2) OR Q=0 THEN 1930
1600 DR=INT(RND(0)*4+1)
1610 ON DR GOTO 1620,1700,1780,1860
1620 IF R=0 THEN 1600
1630 M(X,Y)=M(X,Y)+1:X=X+1
1640 I=3*(X-1)
1650 FOR J=3*Y-2 TO 3*Y-1
1660 POKE SCREEN+40*J+I,0
1670 GOSUB 4200
1680 NEXT J
1690 GOTO 1430
1700 IF D=0 THEN 1600
1710 M(X,Y)=M(X,Y)+10:Y=Y+1
1720 J=3*(Y-1)
1730 FOR I=(3*X-2) TO 3*X-1
1740 POKE SCREEN+40*J+I,0
1750 GOSUB 4200
1760 NEXT I
1770 GOTO 1430
1780 IF L=0 THEN 1600
1790 M(X-1,Y)=M(X-1,Y)-1:X=X-1
1800 I=3*X
1810 FOR J=3*Y-2 TO 3*Y-1
1820 POKE SCREEN+40*J+I,0
1830 GOSUB 4200
1840 NEXT J
1850 GOTO 1430
1860 IF U=0 THEN 1600
```

```
1870 M(X,Y-1)=M(X,Y-1)-10:Y=Y-1
1880 J=3*Y
1890 FOR I=3*X-2 TO 3*X-1
1900 POKE SCREEN+40*J+I,0
1910 GOSUB 4200:NEXT I
1920 GOTO 1430
1930 X=INT(RND(1)*13+1)
1940 Y=INT(RND(1)*7+1)
1950 IF M(X,Y)>0 THEN 1930
1960 C=C+1
1970 GOTO 1430
1980 POSITION 10,0
1990 PRINT "THE MAZE IS BUILT";
2000 FOR X=1 TO 20
2010 S1=80
2020 S2=12
2030 SOUND 0,S1,S2,10
2040 FOR Y=1 TO 25
2050 NEXT Y
2060 SOUND 0,0,0,0
2070 FOR Y=1 TO 5
2080 NEXT Y
2090 NEXT X
2100 GRAPHICS 0
2110 FOR X=1 TO 100
2120 NEXT X
2130 FOR X=SCREEN TO SCREEN+840 STEP 840
2140 FOR Y=0 TO 39
2150 POKE X+Y,128
2160 NEXT Y
2170 NEXT X
2180 FOR X=SCREEN TO SCREEN+39 STEP 39
2190 FOR Y=0 TO 840 STEP 40
2200 POKE X+Y,128
2210 NEXT Y
2220 NEXT X
2230 X=1
2240 Y=INT(RND(0)*7+1)
2250 SCR=SCREEN+40*(3*Y-2)+1
2260 POKE SCR,62
2270 POKE SCR+1,62
2280 POKE SCR+40,90
2290 POKE SCR+41,67
```

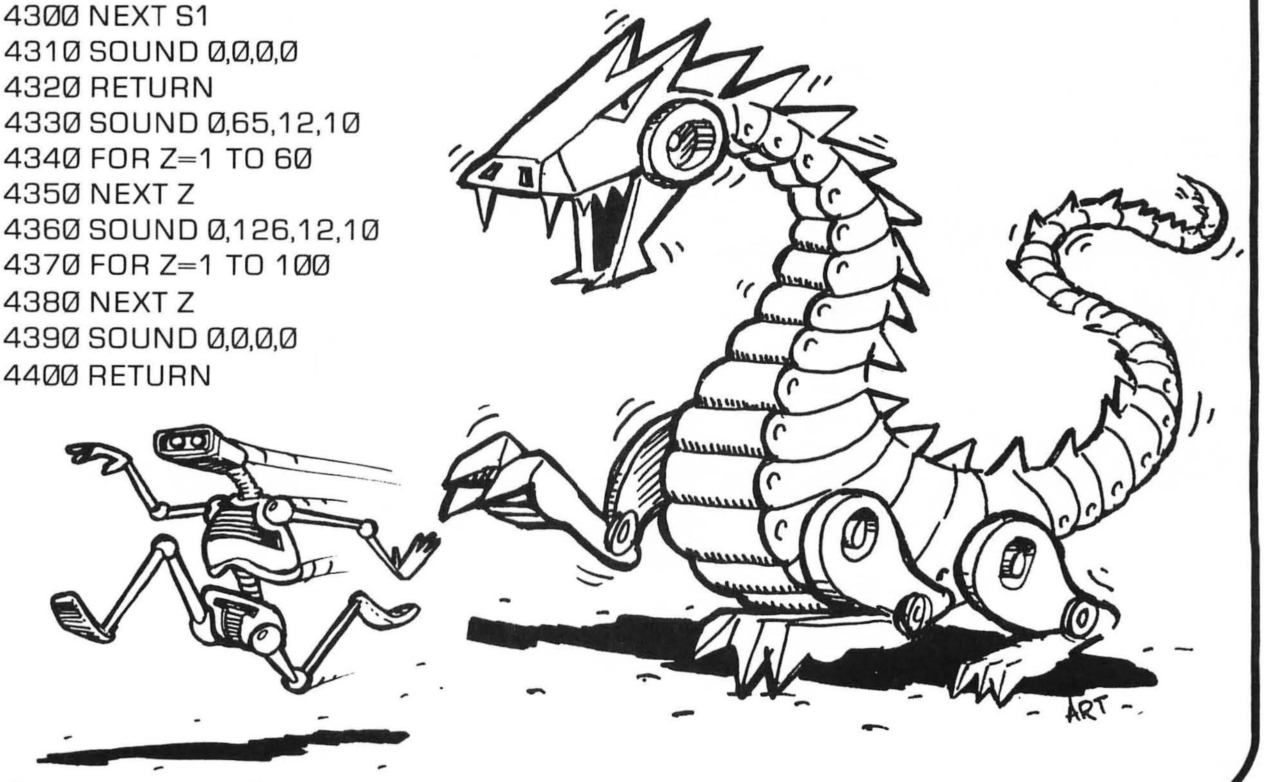
```
2300 HX=3*X-2:HY=3*Y-2
2310 WY=INT(RND(0)*7+1)
2320 POKE SCREEN+40*(3*WY-2)+39,0
2330 REM START DRAGON
2340 SX=13
2350 SY=WY
2360 QY=3*SY-2
2370 RD=1
2380 REM KEYBOARD INPUT
2390 GOSUB 3310
2400 IF SX=X AND SY=Y THEN 3870
2410 IF (OPT=2) OR (OPT=4) THEN 3980
2420 POKE 764,255
2430 K=PEEK(764)
2440 IF K=255 THEN 2430
2450 REM QQ=K GOSUB 2500 K=QQ
2460 REM IF SX=X AND SY=Y THEN 2940
2470 IF K=7 THEN 2530:REM RIGHT
2480 IF K=6 THEN 2840:REM LEFT
2490 IF K=14 THEN 2880:REM UP
2500 IF K=15 THEN 2940:REM DOWN
2510 GOTO 2380
2520 REM YOU MOVE RIGHT
2530 DX=1
2540 DY=0
2550 IF M(X,Y)-10*INT(M(X,Y)/10) THEN 2990
2560 FX=3*X-2
2570 FY=3*Y-2
2580 FOR I=1 TO 3
2590 FX=FX+DX
2600 FY=FY+DY
2610 FOR K=0 TO 1:FOR L=0 TO 1
2620 POKE SCREEN+40*(HY+L)+(HX+K),0
2630 NEXT L
2640 NEXT K
2650 SCR=SCREEN+40*FY+FX
2660 S=S-1
2670 POKE SCR,62
2680 POKE SCR+1,62
2690 POKE SCR+40,90
2700 POKE SCR+41,67
2710 HX=FX
2720 HY=FY
2730 S1=INT(RND(0)*40)+15
```

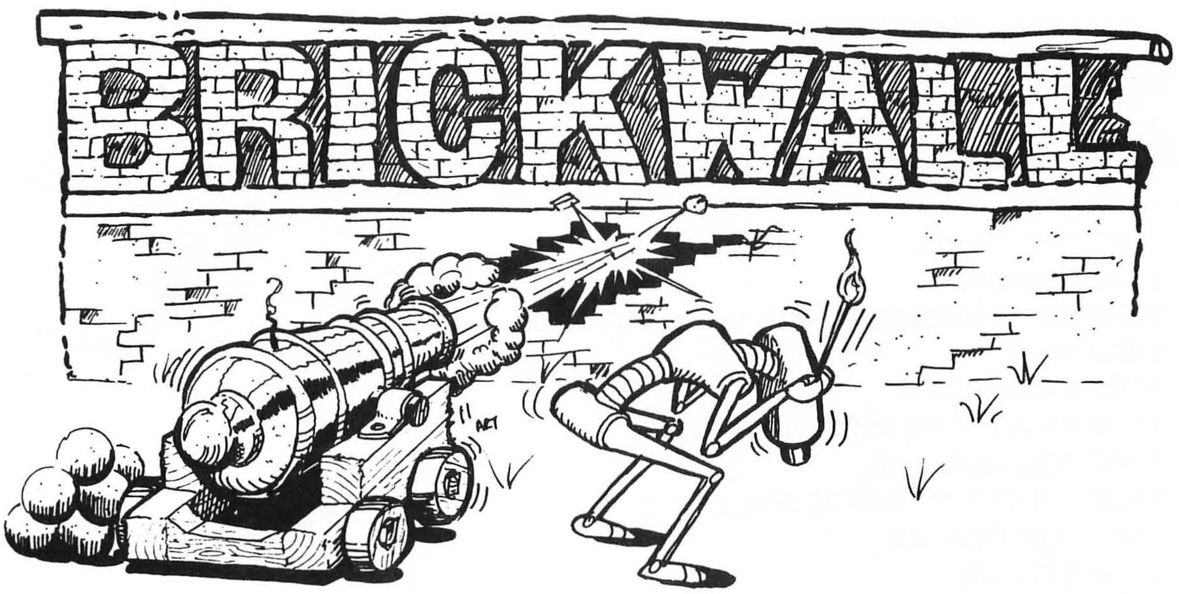
```
2740 SOUND 0,S1,10,10
2750 FOR Z=1 TO 6
2760 NEXT Z
2770 SOUND 0,0,0,0
2780 NEXT I
2790 X=X+DX
2800 Y=Y+DY
2810 IF X=13 AND Y=WY THEN 3190
2820 GOTO 2380
2830 REM YOU MOVE LEFT
2840 DX=-1
2850 DY=0
2860 IF M(X-1,Y)-10*INT(M(X-1,Y)/10)< >0 THEN 2990
2870 GOTO 2560
2880 REM UP
2890 IF Y=1 THEN 2990
2900 IF M(X,Y-1)<-9 THEN 2990
2910 DX=0
2920 DY=-1
2930 GOTO 2560
2940 REM DOWN
2950 DX=0
2960 DY=1
2970 IF M(X,Y)<-9 THEN 2990
2980 GOTO 2560
2990 REM BLOCKAGE
3000 OG=(SCREEN-123)+3*X+120*Y
3010 IF K=15 THEN GOSUB 3140
3020 IF K=6 THEN GOSUB 3120
3030 IF K=7 THEN GOSUB 3100
3040 IF K=14 THEN GOSUB 3160
3050 FOR I=0 TO 3
3060 POKE OG+I*DEL,96
3070 NEXT I
3080 GOSUB 4280
3090 GOTO 2380
3100 REM RIGHT
3110 OG=OG+3
3120 DEL=40
3130 RETURN
3140 REM DOWN
3150 OG=OG+120
3160 REM UP
```

```
3170 DEL=1
3180 RETURN
3190 REM YOU WIN
3200 S=S+20
3210 GRAPHICS 18
3220 PRINT #6
3230 PRINT #6
3240 PRINT #6;"  YOU WIN"
3250 PRINT #6
3260 PRINT #6;"  SCORE = ";S
3270 POKE 764,255
3280 IF PEEK(764)< >255 THEN 1190
3290 IF STRIG(0)=0 THEN 1190
3300 GOTO 3280
3310 REM DRAGON MOVES
3320 IF (OPT=3) OR (OPT=4) THEN 4050
3330 IF X>SX THEN 3370
3340 IF Y>SY THEN 3690
3350 IF X<SX THEN 3750
3360 IF Y<SY THEN 3810
3370 IF SX=13 THEN 3690
3380 IF T(SY,SY)>9 THEN 3400
3390 IF M(SX,SY)-10*INT(M(SX,SY)/10) THEN 3690
3400 DX=1
3410 DY=0
3420 RX=3*SX-2
3430 RY=3*SY-2
3440 FOR I=1 TO 3
3450 RX=RX+DX
3460 RY=RY+DY
3470 FOR K=0 TO 1
3480 FOR L=0 TO 1
3490 POKE SCREEN+40*(QY+L)+(QX+K),0
3500 NEXT L
3510 NEXT K
3520 SCR=SCREEN+40*RY+RX
3530 POKE SCR,190
3540 POKE SCR+1,190
3550 POKE SCR+40,209
3560 POKE SCR+41,197
3570 QX=RX
3580 QY=RY
3590 S1=INT(RND(0)*20)+20
3600 SOUND 0,S1,12,10
```

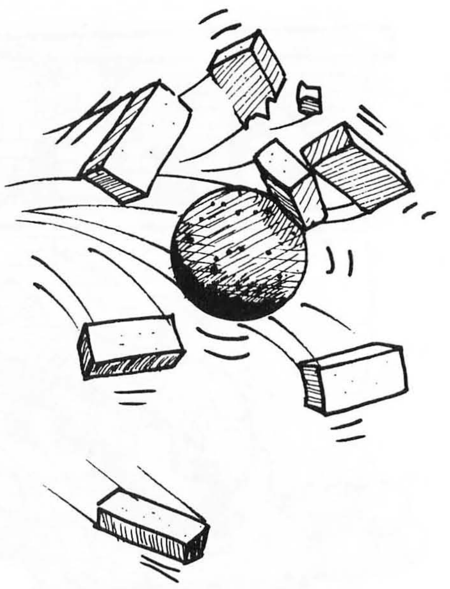
```
3610 FOR Z=1 TO 6
3620 NEXT Z
3630 SOUND 0,0,0,0
3640 NEXT I
3650 SX=SX+DX
3660 SY=SY+DY
3670 T(SX,SY)=T(SX,SY)+1
3680 RETURN
3690 IF SY=7 THEN 3750
3700 IF T(SX,SY)>9 THEN 3720
3710 IF M(SX,SY)<-9 THEN 3750
3720 DX=0
3730 DY=1
3740 GOTO 3420
3750 IF SX=1 THEN 3810
3760 IF T(SX,SY)>9 THEN 3780
3770 IF M(SX-1,SY)-10*INT(M(SX-1,SY)/10) THEN 3810
3780 DX=-1
3790 DY=0
3800 GOTO 3420
3810 IF SY=1 THEN 3370
3820 IF T(SX,SY)>9 THEN 3840
3830 IF M(SX,SY-1)<-9 THEN 3370
3840 DX=0
3850 DY=-1
3860 GOTO 3420
3870 REM DONT KNOW THIS STMT
3880 GRAPHICS 18
3890 PRINT #6
3900 PRINT #6
3910 PRINT #6;" THE DRAGON GOT YOU!"
3920 GOSUB 4330
3930 S=S-20
3940 PRINT #6
3950 PRINT #6
3960 PRINT #6;" SCORE = ";S
3970 GOTO 3270
3980 REM PLAYER JOYSTICK
3990 IF STICK(0)=15 THEN 3990
4000 IF STICK(0)=14 THEN K=14:GOTO 2450
4010 IF STICK(0)=7 THEN K=7:GOTO 2450
4020 IF STICK(0)=13 THEN K=15:GOTO 2450
4030 IF STICK(0)=11 THEN K=6:GOTO 2450
4040 GOTO 3980
```

```
4050 REM 2 PLAYER
4060 IF OPT=3 THEN 4130
4070 IF STICK(1)=15 THEN 4070
4080 IF STICK(1)=14 THEN KK=14:GOTO 3810
4090 IF STICK(1)=11 THEN KK=6:GOTO 3750
4100 IF STICK(1)=13 THEN KK=15:GOTO 3690
4110 IF STICK(1)=7 THEN KK=7:GOTO 3370
4120 GOTO 4070
4130 POKE 764,255
4140 KK=PEEK(764):IF KK=255 THEN 4140
4150 IF KK=7 THEN 3370
4160 IF KK=15 THEN 3690
4170 IF KK=6 THEN 3750
4180 IF KK=14 THEN 3810
4190 GOTO 4060
4200 S1=INT(RND(0)*200+10)
4210 S2=10
4220 IF RND(0)>0.5 THEN S2=12
4230 SOUND 0,S1,S2,10
4240 FOR Z=1 TO 25
4250 NEXT Z
4260 SOUND 0,0,0,0
4270 RETURN
4280 SOUND 0,70,12,10
4290 FOR S1=1 TO 25
4300 NEXT S1
4310 SOUND 0,0,0,0
4320 RETURN
4330 SOUND 0,65,12,10
4340 FOR Z=1 TO 60
4350 NEXT Z
4360 SOUND 0,126,12,10
4370 FOR Z=1 TO 100
4380 NEXT Z
4390 SOUND 0,0,0,0
4400 RETURN
```



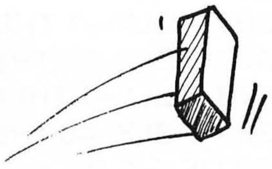


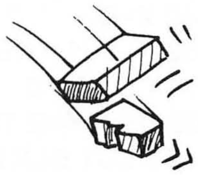
This version of Brick Wall has a fiendish capability that allows you to move up on the ball and bat it back at short range. You give up some maneuverability, but you can rack up a humongous score when you have mastered the technique. Once you have penetrated the wall a domino effect will increase the number of bricks that are knocked out at each stroke. When there are only a few bricks left the skill level required goes up dramatically. The paddle routine in 1700 to 1760 is very forgiving. It reads upper left, center left, and lower left and includes them all as an instruction to move left. The same pattern holds for all four directions, yielding smooth diagonal movements if desired. The program has a demo mode that is invoked by changing line 1070 to DEMO=1. It is a bit discouraging watching the computer, because it never misses. Line 1810 introduces a random factor in the direction the ball bounces, and the demo mode skips this line, so you can say the demo mode cheats a little bit. If the game is getting the best of you, give yourself a few more balls in line 1430. The last few bricks are hard to hit unless you put some "English" on the ball. Hitting the ball with the left corner of the paddle sends the ball to the left and the right corner sends the ball to the right. You might arbitrarily declare a score of 5000 as a win and build in some bells and whistles when this score is reached.



```
1000 REM  ************************************
1010 REM *                                  *
1020 REM *           BRICK WALL             *
1030 REM *                                  *
1040 REM  ************************************
1050 REM
1060 DIM A$(20)
1070 DEMO=0
1080 GOTO 1200:REM SKIP SUBROUTINES
1090 REM
1100 GRAPHICS 0
1110 PRINT "}":REM CLEAR SCREEN
1120 POSITION 10,6
1130 PRINT "**** BRICK WALL ****"
1140 POSITION 2,8
1150 RETURN
1160 POSITION 2,23
1170 PRINT "PRESS 'RETURN' TO CONTINUE";
1180 INPUT A$
1190 RETURN
1200 GOSUB 1100
1210 PRINT "IN THIS GAME YOU WILL SEE A SIDE VIEW"
1220 PRINT "OF A BRICK WALL, AND A PADDLE"
1230 PRINT "THE BRICK WALL WILL BE AT THE BOTTOM"
1240 PRINT "OF THE SCREEN AND THE PADDLE AT THE"
1250 PRINT "TOP."
1260 PRINT
1270 PRINT " YOU START WITH THREE BALLS AND"
1280 PRINT "MUST BREAK UP THE WHOLE WALL TO WIN."
1290 PRINT "MOVE THE PADDLE LEFT AND RIGHT WITH"
1300 PRINT "THE JOYSTICK. START THE NEXT BALL"
1310 PRINT "WITH THE FIRE BUTTON"
1320 GOSUB 1160
1330 HS=0:REM HIGH SCORE
1340 GRAPHICS 5
1350 SC=0
1360 FOR Y=0 TO 3
1370 FOR X=0 TO 79
1380 COLOR (X+Y-2*INT((X+Y)/2)+1)
1390 PLOT X,2*Y+20
1400 PLOT 79-X,2*(7-Y)+20
1410 NEXT X
1420 NEXT Y
1430 BL=3
```

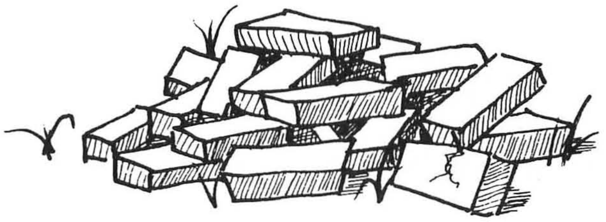
```
1440 POKE 752,127
1450 PRINT "}HIGH SCORE ";HS
1460 PRINT "YOUR SCORE ";SC
1470 PRINT
1480 PRINT "BALLS LEFT ";BL;
1490 REM
1500 PY=3:REM PADDLE Y
1510 PX=39
1520 BLX=PX:REM BALL X SARTS CENTER
1530 BLY=PY+1:REM BALL Y STARTS LINE 2
1540 BRK=8*80:REM NUMBER OF BRICKS LEFT
1550 SC=0:REM SCORE OF CURRENT GAME
1560 COLOR 2
1570 PLOT PX-3,PY:DRAWTO PX+3,PY:REM THE PADDLE FIRST TIME
1580 COLOR 5
1590 DX=1:REM X START DIRECTION
1600 DY=1:REM Y START DIRECTION
1610 REM
1620 REM TOP OF PLAY LOOP
1630 REM
1640 IF STRIG(0)=1 AND DEMO=0 THEN 1640
1650 OX=BLX
1660 OY=BLY
1670 BLX=BLX+DX
1680 BLY=BLY+DY
1690 OPX=PX-3:OPY=PY
1700 K=STICK(0)
1710 U=0
1720 IF K=14 OR K=10 OR K=6 THEN U=-1
1730 IF K=13 OR K=9 OR K=5 THEN U=1
1740 W=0
1750 IF K=7 OR K=6 OR K=5 THEN W=1
1760 IF K=11 OR K=10 OR K=9 THEN W=-1
1770 IF DEMO=0 THEN 1820
1780 IF PX>BLX THEN W=-1
1790 IF PX<BLX THEN W=1
1800 U=1
1810 IF INT(RND(0)+0.5)=1 THEN U=-1
1820 PX=PX+2*W
1830 PY=PY+U
1840 IF PY<3 THEN PY=3
1850 IF PY>18 THEN PY=18
1860 IF PX<3 THEN PX=3
1870 IF PX>76 THEN PX=76
```





```
1880 X=0:Y=0
1890 IF BLX=-1 THEN BLX=0:DX=1:Y=1
1900 IF BLX=80 THEN BLX=79:DX=-1:Y=1
1910 IF BLY=1 THEN BLY=2:DY=1:X=1
1920 IF BLY=40 THEN BLY=39:DY=-1:X=1
1930 IF Y=1 OR X=1 THEN SOUND 1,150,10,10
1940 IF OPX+3=PX AND OPY=PY THEN 1970
1950 COLOR 0
1960 PLOT OPX,OPY:DRAWTO OPX+6,OPY
1970 COLOR 2
1980 PLOT PX-3,PY:DRAWTO PX+3,PY
1990 IF BLY=PY THEN 2020
2000 IF OY=PY THEN 2020
2010 GOTO 2120
2020 IF BLX<PX-3 THEN 2120
2030 IF BLX>PX+3 THEN 2120
2040 SOUND 1,20,10,10
2050 BLX=OX
2060 BLY=PY+1
2070 DY=1
2080 DX=0
2090 IF BLX<PX-1 THEN DX=-1
2100 IF BLX>PX+1 THEN DX=1
2110 GOTO 2280
2120 LOCATE BLX,BLY,K
2130 IF K=0 OR K=3 THEN 2280
2140 SOUND 1,10,10,10
2150 Y=INT((BLY-20)/4)
2160 SC=SC+1*(Y=0)+Y*10*(Y>0)
2170 BRK=BRK-1
2180 IF BRK=0 THEN 2500
2190 IF SC>HS THEN HS=SC
2200 POKE 656,0:POKE 657,13
2210 POKE 752,127
2220 PRINT HS;
2230 POKE 656,1:POKE 657,13
2240 PRINT SC;
2250 DY=DY-DY*2
2260 DX=INT(3*RND(0))-1
2270 X=0
2280 IF X=1 THEN 2260
2290 COLOR 3
2300 IF BLY=2 THEN 2360
2310 PLOT BLX,BLY
```

```
2320 COLOR 0
2330 PLOT OX,OY
2340 SOUND 1,0,0,0
2350 GOTO 1650
2360 SOUND 1,100,12,10
2370 FOR X=1 TO 100:NEXT X
2380 BL=BL-1
2390 IF BL=0 THEN 2470
2400 COLOR 0
2410 PLOT OX,OY
2420 BLX=PX
2430 BLY=PY+1
2440 DY=1:DX=0
2450 SOUND 1,0,0,0
2460 GOTO 1640
2470 PRINT "} YOU LOST YOUR LAST BALL"
2480 PRINT "BUT YOUR SCORE IS ";SC
2490 GOTO 2520
2500 PRINT "}YOU GOT ALL OF THE BRICKS"
2510 PRINT "CONGRATULATIONS A GREAT GAME"
2520 IF SC<HS THEN 2540
2530 PRINT "A NEW HIGH SCORE !!!!"
2540 PRINT "YOUR SCORE WAS ";SC;". AGAIN Y/N";
2550 SOUND 1,0,0,0
2560 IF DEMO=1 THEN A$="Y":GOTO 2580
2570 INPUT A$
2580 IF A$="Y" THEN 1340
```



Land Baron

This is a version of a popular board game that allows you to become the real estate czar of Atlantic City. Two to four players can participate. There are lots of DATA and PRINT statements, but the program is relatively simple. The program looks for 1-4 different joysticks in 1570 and 1580. If you only have one stick you can change the variable to 1 and pass the joystick around. The Atari graphic characters do not print out on most printers, and the screen display does not indicate which keys must be pressed to generate the desired graphic. We have substituted a modified listing that shows all the control characters as bracketed letters. [H] indicates that you are to hold down the control key while you press H. A white triangle should appear when you do this. Not only will these characters not appear on the printed page, but some printers will interpret the contol charcters as orders to print special sizes of type or even form feed the paper. You can check your programming, a replica of the game board will be displayed in the listings on the TV screen, just as it appears when the game plays.

```
1ØØØ OPEN #1,6,Ø,"K:":OPEN #2,4,Ø,"S:" 1Ø1Ø REM APR 24 18:ØØ 1Ø2Ø
     PRINT "}":REM CLEAR SCREEN
1Ø3Ø DIM X$(85),M(4),P(4),N1$(6),N2$(6),N3$(6),N4$(6),Y$(11),Q$(7),
     T$(1),TK$(4),TL$(7),P$(4Ø),OL$(1),N$(6)
1Ø4Ø SETCOLOR 2,34,1:SETCOLOR 4,34,1:SETCOLOR 1,1,8:POKE 752,3:POKE
     82,1:POKE 83,39:POKE 77,Ø
1Ø5Ø TL$="[,][.][;][P]":REM INVERSE CONTROL CHARACTERS
```

These characters are an inverse club, spade, diamond and heart

```
1Ø6Ø
P$="ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ9"
1Ø7Ø FOR I=Ø TO 23:READ X$:POSITION 1,I:PRINT X$;:NEXT I
```

Write all the line numbers and DATA statements first in normal characters. Press the ATARI key for the inverse characters.

> are normal spaces.
{ } empty braces are inverse spaces.
{R} braces are inverse letters.
[U] brackets are control characters.
(:) parentheses are normal characters.
!Q! exclamations are inverse control characters.

The game board will appear in the listing as it will appear on the screen.

```
1Ø8Ø DATA
[U][U][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][U][O]
1Ø9Ø DATA
{ }{ }{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Z![Y]
```

Copy line 1090 here and edit for changes.

```
11ØØ DATA
{ }{ }{ }!Y!{ }!Y!{ }!Y!{?}!Y!{ }!Y!{ }!Y!{R}!Y!{ }!Y!{ }!Y!{ }!Y![Z]{ }[Y]
111Ø DATA
[U][U][U][Q][R][R][R][R][R][R][R][R][R][R][R][E] > > > > > [I][U][U][O]
112Ø DATA
{ }{ }{ }(:) > > > > > > > > > > > (:) > [H][J] > > !Y!{ }{ }[Y]
113Ø DATA
[U][U][U](:) > L A N D L O R D > > (:)[H]{ }{ }[J] > [I][U][U][O]
```

Copy line 1110 here and edit for changes.

```
114Ø DATA
[U][U][U](:) > > > > > > > > > > > (:)!J!{ }{?}{ }[J][I][U][U][O]
115Ø DATA
[U][U][U][A][R][R][R][R][R][R][R][R][R][R][R][D] > !J!{ }{ }!H![I][U][U][O]
116Ø DATA
{ }{C}{ }(:) > > > > > > > > > > > (:) > > !J!!H! > !Y!{ }{C}[Y] 117Ø DATA
[U][U][U](:) > > > > > > > > > > > (:) > > > > > [I][U][U][O]
```

Copy this line as needed below.

```
1180 DATA
{ }{ }{ }(:) > > > > > > > > > > > (:) > > > > > !Y!{ }{ }[Y]
```

Copy this for every other line.

```
1190 DATA
[U][U][U](:) > > > > > > > > > > > (:) > > > > > [I][U][U][O]
1200 DATA
{ }{R}{ }(:) > > > > > > > > > > > (:) > > > > > !Y!{R}{ }[Y]
1210 DATA
[U][U][U](:) > > > > > > > > > > > (:) > > > > > [I][U][U][O]
1220 DATA
{ }{ }{ }(:) > > > > > > > > > > > (:) > > > > > !Y!{?}{ }[Y]
1230 DATA
[U][U][U][Z][R][R][R][R][R][R][R][R][R][R][R][C] > > > > > [I][U][U][O]
1240 DATA
{ }{ }{ } > > [H][J] > > > > > > > > > > > > > > !Y!{ }{ }[Y]
1250 DATA
[U][U][U] > [H]{ }{ }[J] > > > > > > > > > > > > > [I][U][U][O]
1260 DATA
{ }{ }{ } > !J!{ }{C}{ }[J] > > > > > > > > > > > > > !Y!{ }{C}[Y]
1270 DATA
[U][U][U] > > !J!{ }{ }!H! > > > > > > > > > > > > [I][U][U][O]
1280 DATA
{ }{ }{ } > > > !J!!H! > > > > > > > > > > > > > !Y!{ }{ }[Y]
1290 DATA
[U][U][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][I][U][U][O]
1300 DATA
(:)(:)(:)!Y!{ }!Y!{ }!Y!{?}!Y!{ }!Y!{R}!Y!{ }!Y!{ }!Y!{C}!Y!{ }!Y!{G}{O}[Y]
1310 DATA
{ }{ }{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }!Y!{ }{<}[Y]
```

```
1320 TK$="[P][;][.][,]":REM INVERSE CONTROL CHARACTERS
1330 FOR T=1 TO 4:POSITION 3Ø,1Ø:? "WHO ISPLAYER #";T;:S=12:GOSUB
     176Ø:IF X=5 THEN 141Ø
134Ø M(T)=15ØØ:P(T)=Ø
135Ø ON T GOTO 136Ø,137Ø,138Ø,139Ø
136Ø N1$=Q$:GOTO 14ØØ
137Ø N2$=Q$:GOTO 14ØØ
138Ø N3$=Q$:GOTO 14ØØ
139Ø N4$=Q$
14ØØ POSITION 1Ø,16+T:? Q$:POSITION 15,16+T:? CHR$(ASC(TK$(T,T))-128)
     :PLY=T:NEXT T
141Ø IF T<3 THEN T=T-1:X=Ø:NEXT T
142Ø FOR T=1 TO PLY:GOSUB 172Ø:F=Ø:ON T GOTO 143Ø,144Ø,145Ø,146Ø
143Ø N$=N1$:GOTO 147Ø
144Ø N$=N2$:GOTO 147Ø
145Ø N$=N3$:GOTO 147Ø
146Ø N$=N4$
147Ø POSITION 28,1Ø:? N$;" 'S";:POSITION 29,11:? "TURN";
148Ø GOSUB 153Ø:TRØ:OLD=P(T):GOSUB 184Ø:IF RN=D1 THEN GOSUB
     174Ø:IF F>2 THEN NEXT T
149Ø GOSUB 224Ø:P(T)=R:IF TR=1 THEN 148Ø
15ØØ NEXT T
151Ø GOSUB 172Ø:POSITION 27,13:? " PUSH
     JYSTKBUTTONFORPROPERTYLIST";:FOR I=1 TO 3ØØ:NEXT I
152Ø POP :POKE 77,Ø:GOTO 142Ø
153Ø FOR I=1 TO PLY:POSITION 17,16+I:? " ";M(I);:IF M(I)<Ø THEN GOSUB 18ØØ
154Ø NEXT I
155Ø RETURN
156Ø POSITION 27,2Ø:? "PUSH JOYSTKUP YESDOWN NO";:IF STRIG(T-1)=15
     THEN GOSUB 338Ø
157Ø IF STRIG(T-1)=Ø THEN GOSUB 338Ø:GOSUB 225Ø:RETURN
158Ø SK=STICK(T-1):IF SK=14 OR SK=13 THEN RETURN
159Ø GOTO 156Ø
16ØØ POSITION 27,2Ø:? "HIT JOYSTK";:IF STICK(T-1)=15 THEN 16ØØ
161Ø RETURN
162Ø POSITION X,Y:PRINT ".":RETURN
163Ø POSITION X,Y:PRINT "..":RETURN
164Ø POSITION X,Y:PRINT "...":RETURN
165Ø POSITION X,Y:PRINT ". .. .":RETURN
166Ø POSITION X,Y:PRINT ". ... .":RETURN
167Ø POSITION X,Y:PRINT "......":RETURN
168Ø POSITION X,Y:PRINT "          ":RETURN
169Ø POSITION X,Z:PRINT "                  ":RETURN
```

```
1700 RESTORE :FOR I=0 TO 23:READ X$:NEXT I
1710 FOR I=0 TO R-1:READ X$,X$,X$,X$:NEXT I:RETURN
1720 FOR I=1 TO 13:POSITION 26,I+9:? "             ";:NEXT I:RETURN
1730 FOR I=1 TO 11:POSITION 26,I+11:? "             ";:NEXT I:RETURN
1740 F=F+1:POSITION 27,16:? "YOU ROLLEDDOUBLES";:IF F=3 THEN ?
     "3 TIMES";:R=30:P(T)=10:GOSUB 2240
1750 TR=1:RETURN
1760 POSITION 30,S:? "?         ";:Q$="":FOR I=1 TO 5:GET #1,IN:IF IN=155
     THEN 1780
1770 ? CHR$(IN);:Q$(I,I)=CHR$(IN):NEXT I
1780 IF Q$="" THEN Q$="P":X=5
1790 PRINT " ":RETURN
1800 REM BANKRUPT
1810 GOSUB 1720:IF M(1)<0 THEN POSITION 30,13:? N1$;:GOTO 1830
1820 IF M(2)<0 THEN POSITION 30,13:? N2$;
1830 POSITION 26,15:? " YOURBANKRUPT";:END
1840 GOSUB 1730:POSITION 27,13:? "ROLL WITH":POSITION 27,14:PRINT
     "JOYSTICK ";T
1850 IF STRIG(T-1)=0 THEN GOSUB 3380
1860 IF STICK(T-1)=15 THEN 1840
1870 GOSUB 1730
1880 GOSUB 3210:Y=RN:X=27:Z=O1Y:GOSUB 1690:GOSUB 1680:X=28:ON
     RN GOSUB 1620,1630,1640,1650,1660,1670:O1Y=Y
1890 D1=RN
1900 GOSUB 3210:Y=RN:X=32:Z=O2Y:GOSUB 1690:GOSUB 1680:X=33:ON
     RN GOSUB 1620,1630,1640,1650,1660,1670:O2Y=Y
1910 ROLL=RN+D1:R=ROLL+OLD:IF R<41 THEN 1930
1920 R=R-40:M(T)=M(T)+200:GOSUB 1530
1930 OLD=R:RETURN
1940 FOR Z=0 TO 10:POSITION 4,Z+4:PRINT ":             :":NEXT Z:READ
     X$,PRICE,RENT:POSITION 5,6:? X$
1950 IF Y$="ST" THEN 32767
1960 IF Y$="CC" THEN GOSUB 2420
1970 IF Y$="C" THEN GOSUB 2550
1980 IF Y$="T" AND M(T)>1999 THEN M(T)=M(T)-200:GOTO 2050
1990 IF Y$="T" THEN M(T)=M(T)-INT(M(T)*0.1)
2000 IF Y$="R" THEN GOSUB 2300
2010 IF Y$="U" THEN GOSUB 2300
2020 IF Y$="J" THEN R=10:P(T)=10:TR=0:M(T)=M(T)-50:GOTO 2240
2030 IF Y$="L" THEN M(T)=M(T)-75
2040 IF Y$="G" THEN M(T)=M(T)+200:R=0
2050 Y$="ST":RETURN
2060 POSITION 23,1:? " ";:WK=R:R=P(T):SD=1:GOSUB 2120:GOSUB 2190
2070 VK=0:IF WK<P(T) THEN VK=40
2080 TE=1:IF BACK=1 THEN TE=-1:BACK=0:VK=0
```

```
2090 FOR R=P(T)+1 TO (WK+VK) STEP TE:GOSUB 2120:GOSUB 2190:NEXT
     R:R=WK
2100 GOSUB 2120:PRINT TK$(T,T);
2110 P(T)=R:RETURN
2120 DF=R:IF R>40 THEN R=R-40
2130 IF R=0 OR R=40 THEN X=23:Y=23
2140 IF R>0 AND R<11 THEN Y=23:X=23-(R*2)
2150 IF R>10 AND R<21 THEN X=1:Y=22-((R-10)*2)
2160 IF R>20 AND R<31 THEN Y=1:X=(R-20)*2+3
2170 IF R>30 AND R<40 THEN X=24:Y=(R-30)*2+2
2180 POSITION X,Y:R=DF:RETURN
2190 POSITION X,Y:GET #2,UY:POSITION X,Y:IF SD=1 THEN SD=0:GOTO 2210
2200 PRINT TK$(T,T);:SOUND 2,100,100,100:SOUND 2,0,0,0
2210 IF CHR$(UY)=TK$(T,T) THEN UY=160
```

```
2220 POSITION X,Y:PRINT CHR$(UY);
2230 FOR Z9=1 TO 10:NEXT Z9:RETURN
2240 GOSUB 2060
2250 GOSUB 1700:READ Y$:IF LEN(Y$)<3 THEN GOTO 1940
2260 FOR Z=0 TO 10:POSITION 5,Z+4:PRINT "            ":NEXT Z
2270 POSITION 4,7:? "":POSITION 5,5:PRINT Y$
2280 READ X$:POSITION 5,8:PRINT X$
2290 READ PRICE,RENT:POSITION 6,12:PRINT "PRICE $";PRICE
2300 GOSUB 1530:GOSUB 1730:PR=VAL(P$(R,R)):IF PR=T THEN 2400
2310 IF PR=0 THEN 2380
2320 IF PR>5 THEN STOP
2330 POSITION 5,4:? CHR$(ASC(TK$(PR))-128);
2340 IF Y$="R" THEN GOSUB 2710
2350 IF Y$="U" THEN GOSUB 2780
2360 POSITION 28,13:? "YOU OWERENT TOPLAYER";PR;"$";RENT;
```

```
      :M(T)=M(T)-RENT:M(PR)=M(PR)+RENT
2370 GOSUB 1600:RETURN
2380 POSITION 28,13:? "DO YOUWANT TOBUY IT";:GOSUB 1560:IF
      SK=13 THEN RETURN
2390 M(T)=M(T)-PRICE:P$(R,R)=STR$(T):RETURN
2400 POSITION 5,4:? CHR$(ASC(TK$(PR))-128);
2410 POSITION 29,13:? "YOU OWNIT":FOR I=1 TO 150:NEXT I:RETURN
2420 RH=RH+1:POSITION 5,10:ON RH GOSUB 2450,2460,2470,2480,2490,
      2500,2510,2520,2530,2540
2430 IF RH=10 THEN RH=0
2440 M(T)=M(T)+M:M=0:RETURN
2450 ? " XMAS FUNDMATURESCOLLECT$100":M=100:RETURN
2460 ? "LIFEINSURANCEMATURESCOLLECT$100":M=100:RETURN
2470 ? "INCOME TAXREFUNDCOLLECT$20":M=20:RETURN
2480 ? "DOCTOR'SFEEPAY$50":M=-50:RETURN
2490 ? "YOUINHERIT$100":M=100:RETURN
2500 GOTO 2640:REM GO
2510 ? "PAYHOSPITAL$100":M=-100:RETURN
2520 GOTO 2670:REM JAIL
2530 ? "PAYSCHOOL TAX$50":M=-50:RETURN
2540 ? "BANKERRORCOLLECT$200":M=200:RETURN
2550 RC=RC+1:POSITION 5,10:ON RC GOTO 2590,2600,2630,2640,2650,
      2660,2670,2680,2690,2700,2560
2560 RC=0:GOTO 2550
2570 POSITION 27,20:? "HIT JOYSTK";:IF STICK(T-1)=15 THEN 2570
2580 GOSUB 1730:GOTO 2240
2590 ? "PAY POORTAX OF$15":M(T)=M(T)-15:RETURN
2600 ? "ADVANCETOILLINOISAVENUE":O=24:GOTO 2610
2610 P(T)=R:IF O<R THEN M(T)=M(T)+200
```

```
2620 R=O:GOTO 2570
2630 ? "GOBACKTHREESPACES":R=R-3:BACK=1:GOTO 2570
2640 ? "ADVANCETOGO":R=40:GOTO 2570
2650 ? "BANK PAYSDIVIDENDOF $50":M(T)=M(T)+50:RETURN
2660 ? "TAKE A RIDEON THEREADING":O=5:GOTO 2610
2670 ? "GOTOJAIL":R=30:GOTO 2570
2680 ? "BUILDING& LOANMATURESCOLLECT$150":M(T)=M(T)+150:RETURN
2690 ? "ADVANCETOBOARDWALK":O=39:GOTO 2610
2700 ? "ADVANCETOST.CHARLESPLACE":O=11:GOTO 2610
2710 RR=0:IF VAL(P$(5,5))=PR THEN RR=RR+1
2720 IF VAL(P$(15,15))=PR THEN RR=RR+1
2730 IF VAL(P$(25,25))=PR THEN RR=RR+1
2740 IF VAL(P$(35,35))=PR THEN RR=RR+1
2750 IF RR=1 OR RR=2 THEN RENT=RR*25:RETURN
2760 IF RR=3 THEN RENT=100:RETURN
2770 IF RR=4 THEN RENT=200:RETURN
2780 IF VAL(P$(12,12))=VAL(P$(28,28)) THEN RENT=ROLL*10:RETURN
2790 RENT=ROLL*4:RETURN
2800 DATA BLANK,BLANK,0,0
2810 DATA PURPLE,MEDITER-RANEANAVENUE,60,2
2820 DATA CC,COMMUNITYCHEST,0,0
2830 DATA PURPLE,BALTICAVENUE,60,4
2840 DATA T,INCOMETAX'PAY 10%OR$200,200,0
2850 DATA R,READINGRAILROADPRICE $200,200,25
2860 DATA LIGHT BLUE,ORIENTALAVENUE,100,6
2870 DATA C,CHANCE?,0,0
2880 DATA LIGHT BLUE,VERMONTAVENUE,100,6
2890 DATA LIGHT BLUE,CONNECTICUTAVENUE ,120,8
2900 DATA V,JUSTVISITINGJAIL,0,0
2910 DATA MAROON,ST.CHARLESPLACE,140,10
2920 DATA U,ELECTRICCOMPANYPRICE $150,150,0
2930 DATA MAROON,STATESAVENUE,140,10
2940 DATA MAROON,VIRGINIAAVENUE,160,12
2950 DATA R,PENNSYL-VANIARAILROADPRICE $200,200,25
2960 DATA ORANGE,ST.JAMESPLACE,180,14
2970 DATA CC,COMMUNITYCHEST,0,0
2980 DATA ORANGE,TENNESSEEAVENUE,180,14
2990 DATA ORANGE,NEW YORKAVENUE,200,16
3000 DATA F,FREEPARKING,0,0
3010 DATA RED,KENTUCKYAVENUE,220,18
3020 DATA C,CHANCE?,0,0
3030 DATA RED,INDIANAAVENUE,220,18
3040 DATA RED,ILLINOISAVENUE,240,20
3050 DATA R,B.& O.RAILROADPRICE $200,200,25
```

```
3060 DATA YELLOW,ATLANTICAVENUE,260,22
3070 DATA YELLOW,VENTNORAVENUE,260,22
3080 DATA U,WATERWORKSPRICE $150,150,35
3090 DATA YELLOW,MARVINGARDENS,280,24
3100 DATA J,GO TOJAIL,0,0
3110 DATA GREEN,PACIFICAVENUE,300,26
3120 DATA GREEN,NORTHCAROLINAAVENUE,300,26
3130 DATA CC,COMMUNITYCHEST,0,0
3140 DATA GREEN,PENNSYLVANIAAVENUE,320,28
3150 DATA R,SHORTLINEPRICE $200,200,25
3160 DATA C,CHANCE?,0,0
3170 DATA BLUE,PARKPLACE,350,35
3180 DATA L,LUXURYTAXPAY $75,75,0
3190 DATA BLUE,BOARDWALK,400,50
3200 DATA G,GGGGGGGGGGGGGOOOOOOOOOOOOOOO,200,0
3210 RN=INT(RND(0)*6)+1:RETURN
3220 GOSUB 1730:IF STRIG(T-1)=0 THEN 3220:REM EXIT
3230 POSITION 27,13:? "HIT RESETBUTTON TOEXITJOYSTK TOCONTINUE"
3240 IF STRIG(T-1)=0 THEN RUN "D1:MENU"
3250 IF STICK(T-1)=15 THEN 3230
3260 RETURN
3270 RESTORE :FOR I=0 TO 23:READ X$:NEXT I
3280 READ X$,X$,A,B
3290 X=X+1:PL=VAL(P$(X,X)):IF PL=0 THEN READ X$,X$,A,B:GOTO 3290
3300 IF PL=9 THEN GOSUB 1730:RETURN
3310 POSITION 5,4:? CHR$(ASC(TK$(PL))-128);X
3320 READ Y$:IF LEN(Y$)<3 THEN GOTO 3370
3330 FOR Z=1 TO 10:POSITION 5,Z+4:PRINT "            ":NEXT Z
3340 POSITION 4,7:? "":POSITION 5,5:PRINT Y$
3350 READ X$:POSITION 5,8:PRINT X$
3360 READ A,B:POSITION 6,12:PRINT "PRICE $";A:GOSUB 1600:GOTO 3290
3370 FOR Z=1 TO 10:POSITION 4,Z+4:PRINT ":            :":NEXT
     Z:READ X$,A,B:POSITION 5,6:? X$;:GOSUB 1600:GOTO 3290
3380 GOSUB 3220:GOSUB 1730:POSITION 28,13:PRINT
     "PROPERTYOWNED";:X=0:GOTO 3270
```

Atari BASIC is just fast enough to make this game hair-raising and difficult. The screen will contain yellow squares that are capable of electrocuting your player or the robots. The outside frame is also electrically charged, but the robots are too smart to stumble into the frame; only you can do that! The robots are blue, and are attracted to you by electromotive force. If they get to you it's "game over." You win the game by hiding behind the yellow squares and luring the robots to their doom. You can even trick the robots into zapping each other by doubling back behind a yellow square. The title screen has a tricky feature. The PEEKs and POKEs in lines 1090 to 1110 actually change the width of the scan lines on the text screen, and expand the text in a band in the top half of the screen. Press the <BREAK> key to stop the program while the title screen is on. Now, if you type LIST, you will see the program listing scroll by and print out in large type in this area of the screen. The routine in lines 1240 to 1280 inputs the difficulty factor. The variable A gets used later for the size of the steps the robots take. Lines 1290 to 1360 set up a joystick lookup table. When the joystick is read in line 1770 the value for J will be translated into the directions to move by referring to the lookup table. The directions are in the form of +1,-1, and 0's that are loaded from the DATA statements at the end of the program. Lines 1370 to 1430 draw the frame. Lines 1440 to 1510 set up the yellow squares. Lines 1530 to 1610 put the robots on the screen. If the random number routine tries to put the robot on an occupied square the routine rejects that location and runs another random location. Lines 1700 to 1740 change the color of your player while it waits for your move. Lines 1760 to 1890 read the joystick and move the player. Change 1840 to COLOR1 and the player will not be "undrawn" in black. Lines 1900 to 2100 check to see if any robots are left and then moves the robots and detects hits. The locate commands in 2000 to 2030 detect collisions with robots, squares, or border. Lines 2120 to 2190 eliminate the robots that crash into the squares or each other. Finally, the program ends with the parting messages and sound routines.

```
1000 REM  ***********************************
1010 REM *                                  *
1020 REM *          ROBOT CHASE             *
1030 REM *                                  *
1040 REM  ***********************************
1050 REM
1060 DIM JX(10),JY(10),RX(4),RY(4),A$(20)
1070 GRAPHICS 0
1080 SETCOLOR 2,0,0
1090 W=PEEK(741)+256*PEEK(742)+6
1100 POKE W+4,7:REM CHANGE THE
1110 POKE W+5,7:REM DISPLAY LIST
1120 POKE 752,1
1130 POSITION 7,4
1140 PRINT "ROBOT          -CHASE-"
1150 PRINT
1160 PRINT " Object is to get away from the"
1170 PRINT " robots by moving so that squares"
1180 PRINT " are between you and the robots."
1190 PRINT " Do not touch the wall or squares"
1200 PRINT
1210 PRINT " For a new screen press the trigger."
1220 PRINT
1230 PRINT "DIFFICULTY LEVEL (1=HARD,2=EASY)";
1240 INPUT A$
1250 IF A$="" THEN A$="2"
1260 A=INT(VAL(A$))
1270 IF A<1 OR A>2 THEN A=1
1280 A=1/A
1290 FOR I=0 TO 10
1300 READ V
1310 JX(I)=V
1320 NEXT I
1330 FOR I=0 TO 10
1340 READ V
1350 JY(I)=V
1360 NEXT I
1370 GRAPHICS 3
1380 COLOR 2
1390 PLOT 0,0
1400 DRAWTO 39,0
1410 DRAWTO 39,19
1420 DRAWTO 0,19
1430 DRAWTO 0,0
```

```
1440 FOR I=1 TO 10
1450 X=INT(RND(0)*40)
1460 Y=INT(RND(0)*20)
1470 LOCATE X,Y,V
1480 IF V< >0 THEN 1450
1490 COLOR 2
1500 PLOT X,Y
1510 NEXT I
1520 FOR I=1 TO 4
1530 X=INT(RND(0)*40)
1540 Y=INT(RND(0)*20)
1550 LOCATE X,Y,V
1560 IF V< >0 THEN 1530
1570 RX(I)=X
1580 RY(I)=Y
1590 COLOR 3
1600 PLOT X,Y
1610 NEXT I
1620 X=INT(RND(0)*40)
1630 Y=INT(RND(0)*20)
1640 LOCATE X,Y,V
1650 IF V< >0 THEN 1620
1660 PX=X
1670 PY=Y
1680 COLOR 1
1690 PLOT X,Y
1700 F=0
1710 FOR I=0 TO 128
1720 POKE 708,I
1730 IF STICK(0)< >15 THEN I=128:F=1
1740 NEXT I
1750 IF F=0 THEN 1710
1760 POKE 708,40
1770 J=STICK(0)-5
1780 SOUND 0,10,4,8
1790 IF STRIG(0)=0 THEN 2350
1800 X=PX+JX(J)
1810 Y=PY+JY(J)
1820 LOCATE X,Y,V
1830 IF V< >0 THEN 2210
1840 COLOR 0
1850 PLOT PX,PY
1860 COLOR 1
1870 PLOT X,Y
```

```
1880 PX=X
1890 PY=Y
1900 IF (RX(1)+RX(2)+RX(3)+RX(4))<0 THEN PRINT " YOU WIN!!":GOTO 2290
1910 FOR I=1 TO 4
1920 COLOR 0
1930 IF RX(I)<0 THEN 2060
1940 PLOT RX(I),RY(I)
1950 X=RX(I)-PX:Y=RY(I)-PY
1960 IF X<0 THEN RX(I)=RX(I)+A
1970 IF X>0 THEN RX(I)=RX(I)-A
1980 IF Y<0 THEN RY(I)=RY(I)+A
1990 IF Y>0 THEN RY(I)=RY(I)-A
2000 LOCATE RX(I),RY(I),V
2010 IF V=1 THEN 2080
2020 IF V=2 THEN RX(I)=-1:GOTO 2060
2030 IF V=3 THEN 2120
2040 COLOR 3
2050 PLOT RX(I),RY(I)
2060 NEXT I
2070 GOTO 1770
2080 COLOR 1
2090 PLOT PX,PY
2100 PRINT "YOU HAVE BEEN CAUGHT BY A ROBOT!"
2110 GOTO 2240
2120 FOR S=1 TO 4
2130 IF RX(S)=-1 OR S=I THEN 2190
2140 IF RX(S)< >RX(I) OR RY(S)< >RY(I) THEN 2190
2150 COLOR 0
2160 PLOT RX(S),RY(S)
2170 RX(I)=-1
2180 RX(S)=-1
2190 NEXT S
2200 GOTO 2060
2210 IF V=2 THEN PRINT "YOU HAVE BEEN ELECTROCUTED!!!"
2220 IF V=3 THEN PRINT "YOU RAN RIGHT INTO A ROBOT!"
2230 IF V=1 THEN 1840
2240 PRINT "          YOU LOSE."
2250 FOR I=0 TO 128
2260 SOUND 0,I,8,8
2270 POKE 708,I
2280 NEXT I
2290 POKE 708,40
2300 SOUND 0,0,0,0
2310 PRINT
```

```
2320 PRINT "HIT TRIGGER TO PLAY AGAIN!"
2330 IF STRIG(0)< >0 THEN 2330
2340 GOTO 1370
2350 PRINT "GIVE UP? OK HERE'S A NEW SCREEN."
2360 GOTO 2290
2370 DATA 1,1,1,0,-1,-1,-1,0,0,0,0
2380 DATA 1,-1,0,0,1,-1,0,0,1,-1,0
```

Stranded is a two player board game. Each move consists of two parts: you move your piece to an unoccupied square, and then block off one square on the board so that neither player can move onto it. The object of the game is to prevent your opponent from moving. The first player who is unable to move is the loser. The game is written for two joysticks, but you can easily change play to the keyboard arrows by changing the paddle reading subroutine in 1240 to 1310. There is a keyboard arrow reading subroutine in Dragon's Lair. The program has good playing speed because the subroutines are at the beginning of the program. The routine at 1190 to 1230 draws the moving square. The square is drawn in the color set by C and then drawn over in black to make it flicker and move. The grid is set up in 1840 to 2050 and the beginning positions in 2060 to 2090. The play loop starts at 2110. The moves are made by swapping positions from X1 to X2 to X3 as described in Mubble Chase. The subroutine in 1600 sets a flag F and tests the adjacent squares to see if any are empty. If none of the squares are empty the program branches to 2550 to announce the loser.

```
1000 REM ***********************************
1010 REM *                                 *
1020 REM *            STRANDED             *
1030 REM *                                 *
1040 REM ***********************************
1050 REM
1060 DIM A(8,8),A$(20),X(2),Y(2)
1070 GOTO 1710:REM SKIP SUBROUTINES
1080 GRAPHICS 0
1090 PRINT "}"
1100 POSITION 11,3
1110 PRINT "*** STRANDED ***"
1120 POSITION 2,6
1130 RETURN
1140 POSITION 2,23
1150 PRINT "PRESS 'RETURN' TO CONTINUE";
1160 INPUT A$
1170 RETURN
1180 COLOR C
1190 FOR Y1=-1+Y*4 TO 1+Y*4
1200 PLOT 14+X*5,Y1
1210 DRAWTO 17+X*5,Y1
1220 NEXT Y1
1230 RETURN
1240 X1=0
1250 Y1=0
1260 POKE 77,0
1270 IF A=11 OR A=10 OR A=9 THEN X1=-1
1280 IF A=7 OR A=6 OR A=5 THEN X1=1
1290 IF A=14 OR A=10 OR A=6 THEN Y1=-1
1300 IF A=13 OR A=9 OR A=5 THEN Y1=1
1310 RETURN
1320 X2=X(L)+X1
1330 Y2=Y(L)+Y1
1340 IF X2<1 OR Y2<1 THEN RETURN
1350 IF X2>8 OR Y2>8 THEN RETURN
1360 IF ABS(X2-X3)>1 THEN RETURN
1370 IF ABS(Y2-Y3)>1 THEN RETURN
1380 IF A(X2,Y2)>0 THEN RETURN
1390 A(X,Y)=0
1400 X=X2:X(L)=X
1410 Y=Y2:Y(L)=Y
1420 A(X,Y)=L
```

```
1430 RETURN
1440 IF X=X3 AND Y=Y3 THEN RETURN
1450 T=2
1460 RETURN
1470 IF A(X,Y)>0 THEN RETURN
1480 A(X,Y)=3
1490 T=2
1500 RETURN
1510 X2=X
1520 Y2=Y
1530 X2=X2+X1
1540 Y2=Y2+Y1
1550 IF X2<1 OR X2>8 THEN RETURN
1560 IF Y2<1 OR Y2>8 THEN RETURN
1570 IF A(X2,Y2)>0 THEN 1530
1580 X=X2:Y=Y2
1590 RETURN
1600 F=1
1610 X2=-1:IF X=1 THEN X2=0
1620 X3=1:IF X=8 THEN X3=0
1630 Y2=-1:IF Y=1 THEN Y2=0
1640 Y3=1:IF Y=8 THEN Y3=0
1650 FOR X1=X+X2 TO X+X3
1660 FOR Y1=Y+Y2 TO Y+Y3
1670 IF A(X1,Y1)=0 THEN F=0
1680 NEXT Y1
1690 NEXT X1
1700 RETURN
1710 GOSUB 1080
1720 PRINT " THIS IS A GAME FOR TWO PLAYERS."
1730 PRINT "BOTH OF YOU WILL BE PLACED ON AN 8*8"
1740 PRINT "MATRIX. A PLAYER MAY MOVE IN ANY OF"
1750 PRINT "EIGHT DIRECTIONS (PROVIDING THAT THE"
1760 PRINT "DESTINATION IS EMPTY) BY USING THE"
1770 PRINT "JOYSTICK."
1780 PRINT
1790 PRINT " AFTER YOU MOVE, I WILL ASK YOU TO"
1800 PRINT "CHOOSE ON A SECOND SQUARE IN THE"
1810 PRINT "MATRIX. THAT SQUARE WILL THEN BECOME"
1820 PRINT "UNAVAILABLE TO MOVE INTO."
1830 GOSUB 1140
1840 GRAPHICS 5
1850 FOR X=1 TO 8
1860 FOR Y=1 TO 8
```

```
1870 A(X,Y)=0
1880 NEXT Y
1890 NEXT X
1900 A(1,4)=1
1910 A(8,5)=2
1920 COLOR 3
1930 PLOT 18,2
1940 DRAWTO 58,2
1950 DRAWTO 58,34
1960 DRAWTO 18,34
1970 DRAWTO 18,2
1980 FOR Y=6 TO 30 STEP 4
1990 PLOT 18,Y
2000 DRAWTO 58,Y
2010 NEXT Y
2020 FOR X=23 TO 53 STEP 5
2030 PLOT X,2
2040 DRAWTO X,34
2050 NEXT X
2060 X(1)=1:Y(1)=4
2070 X(2)=8:Y(2)=5
2080 C=1:X=1:Y=4:GOSUB 1180
2090 C=2:X=8:Y=5:GOSUB 1180
2100 REM
2110 REM MAIN LOOP FOR PLAY OF GAME
2120 REM
2130 FOR L=1 TO 2
2140 C=L
2150 X=X(L)
2160 Y=Y(L)
2170 GOSUB 1600
2180 IF F=1 THEN 2550
2190 X3=X
2200 Y3=Y
2210 PRINT "} PLAYER # ";L;" IT IS YOUR MOVE"
2220 T=1
2230 C=0
2240 GOSUB 1180
2250 A=STICK(L-1)
2260 IF A< >15 THEN GOSUB 1240:GOSUB 1320
2270 C=L
2280 GOSUB 1180
2290 A=STRIG(L-1)
2300 IF A=0 THEN GOSUB 1440
```

```
2310 T=T+1
2320 ON T GOTO 2230,2220,2325
2325 IF STRIG(A-1)=0 THEN 2325
2330 PRINT "}PLAYER ";L;" PICK A SQUARE TO BLOCK"
2340 FOR X1=1 TO 8
2350 FOR Y1=1 TO 8
2360 IF A(X1,Y1)>0 THEN 2400
2370 X=X1
2380 Y=Y1
2390 X1=8:Y1=X1
2400 NEXT Y1
2410 NEXT X1
2420 T=1
2430 C=0
2440 GOSUB 1180
2450 A=STICK(L-1)
2460 IF A< >15 THEN GOSUB 1240:GOSUB 1510
2470 C=3
2480 GOSUB 1180
2490 A=STRIG(L-1)
2500 IF A=0 THEN GOSUB 1470
2510 T=T+1
2520 ON T GOTO 2430,2420,2530
2530 NEXT L
2540 GOTO 2130
2550 PRINT "}THE GAME IS OVER PLAYER ";L;" LOST"
2560 PRINT
2570 PRINT "WOULD YOU LIKE TO PLAY AGAIN Y/N";
2580 INPUT A$
2590 IF A$="Y" THEN 1560
2600 END
```

POINT ATTACK

Here is a game of PONG in reverse! All you have to do is avoid the little projectiles instead of hitting them. Sound easy? Well bunches of these critters keep pouring out on the screen and bouncing off every surface. The action isn't fast, but the strategy is, because moving the paddle is like moving a barge. You have to plan ahead and figure out the paths of several projectiles at once. Each projectile needs 4 places to store coordinates:

NX = new X position,
NY = new Y position,
LX = last X postion,
LY = last Y position.

In lines 1740 to 1770 the Ls and Ns are swapped until the screen edge is reached, and the projectile bounces back. Lines 1780 and 1790 draw the projectile in black, erasing it before redrawing it in blue in lines 1800 to 1810. Try changing 1780 to color 1 to see the trail of the projectiles. Lines 1860 to 1920 use the LOCATE function to see if the projectile color (3) blue has been drawn over the paddle. If you change 1860 to RETURN, the paddle will not be detected. The screen will completely fill with projectiles. Changing lines around and disconnecting parts of the program is the easiest way to see what line performs what function. Save the correct version to tape or disk before tinkering with the nuts and bolts of the program.

```
1000 REM ***********************************
1010 REM *                                 *
1020 REM *           POINT ATTACK          *
1030 REM *                                 *
1040 REM ***********************************
1050 REM
1060 DIM A$(20),B(3,500)
1070 REM
1080 GOTO 1200:REM SKIP SUBROUTINES
1090 REM
1100 GRAPHICS 0
1110 PRINT "}"
1120 POSITION 9,3
1130 PRINT "*** POINT ATTACK ***"
1140 POSITION 2,7
1150 RETURN
1160 POSITION 2,23
1170 PRINT "PRESS 'RETURN' TO CONTINUE";
1180 INPUT A$
1190 RETURN
1200 GOSUB 1100
1210 PRINT " THIS IS A SIMPLE GAME WHERE YOU"
1220 PRINT "DODGE THE MOVING BLOCKS. YOU GET"
1230 PRINT "POINTS FOR THE NUMBER OF OBJECTS ON"
1240 PRINT "ON THE SCREEN. YOU CAN ONLY MOVE"
1250 PRINT "UP AND DOWN ON THE SCREEN, SO GOOD"
1260 PRINT "LUCK!"
1270 GOSUB 1160
1280 GRAPHICS 3
1290 COLOR 1
1300 PLOT 0,0
1310 DRAWTO 39,0
1320 DRAWTO 39,19
1330 DRAWTO 0,19
1340 DRAWTO 0,0
1350 PRINT "}PRESS FIRE BUTTON TO BEGIN"
1360 X=18
1370 Y=9
1380 N=1
1390 B(0,N)=1:REM STARTING X
1400 B(1,N)=INT(RND(0)*19)+1:REM STARTING Y
1410 B(2,N)=1:REM STARTING X DIRECTION
1420 B(3,N)=2-INT(RND(0)*19)+1:REM STARTING Y DIRECTION
```

```
1430 REM
1440 COLOR 2
1450 PLOT X,Y-1
1460 DRAWTO X,Y+1
1470 GOSUB 1860
1480 IF N=15 THEN 1550
1485 IF INT(RND(0)*100)<90 THEN 1550
1490 N=N+1
1500 B(0,N)=1
1510 B(1,N)=INT(RND(0)*19)+1
1520 B(2,N)=1
1530 B(3,N)=2-INT(RND(0)*19)+1
1540 PRINT "}THERE ARE ";N;" POINTS"
1550 A=STICK(0)
1560 IF A=15 THEN 1690
1570 Y1=Y
1580 IF A=10 OR A=14 OR A=6 THEN Y1=Y1-1
1590 IF A=9 OR A=13 OR A=5 THEN Y1=Y1+1
1600 IF Y1<2 THEN Y1=2
1610 IF Y1>17 THEN Y1=17
1620 COLOR 0
1630 PLOT X,Y-1
1640 DRAWTO X,Y+1
1650 Y=Y1
1660 COLOR 2
1670 PLOT X,Y-1
1680 DRAWTO X,Y+1
1690 FOR LX=1 TO N
1700 OX=B(0,LX)
1710 OY=B(1,LX)
1720 NX=OX+B(2,LX)
1730 NY=OY+B(3,LX)
1740 IF NX<1 THEN NX=1:B(2,LX)=1
1750 IF NX>38 THEN NX=38:B(2,LX)=-1
1760 IF NY<1 THEN NY=1:B(3,LX)=1
1770 IF NY>18 THEN NY=18:B(3,LX)=-1
1780 COLOR 0
1790 PLOT OX,OY
1800 COLOR 3
1810 PLOT NX,NY
1820 B(0,LX)=NX
1830 B(1,LX)=NY
1840 NEXT LX
1850 GOTO 1470
```

```
1860 LOCATE X,Y-1,K
1870 IF K=3 THEN 1930
1880 LOCATE X,Y,K
1890 IF K=3 THEN 1930
1900 LOCATE X,Y+1,K
1910 IF K=3 THEN 1930
1920 RETURN
1930 PRINT "}THE GAME IS OVER"
1940 PRINT
1950 PRINT "THERE WERE ";N;" POINTS"
```

WHAT BETTER WAY TO LEARN PROGRAMMING THAN BY PLAYING GAMES?

In this exciting new book, the authors have compiled a selection of classic Atari BASIC games along with many clever new games. Written in Atari BASIC, they are formatted in a way which lets you adapt any of the routines to your own programs. Each game is explained in easy-to-understand terms — allowing you to modify and customize to your heart's content!

You'll learn principles of text formatting, word games, data statements and input routines. Many of the games will help you to clearly understand how grids are constructed and graphics animated. Simple techniques are given which allow you to dissect Atari BASIC programs and see what makes them tick.

You'll have hours of fun learning to program in this easy, enjoyable way. And when you're done, you'll have the games to play! Whoever thought that gaming could teach so much about programming and debugging?

ISBN 0-88190-118-0

0 48831 00118

DATAMOST™ INC

8943 Fullbright Avenue, Chatsworth, CA 91311-2750
(818) 709-1202