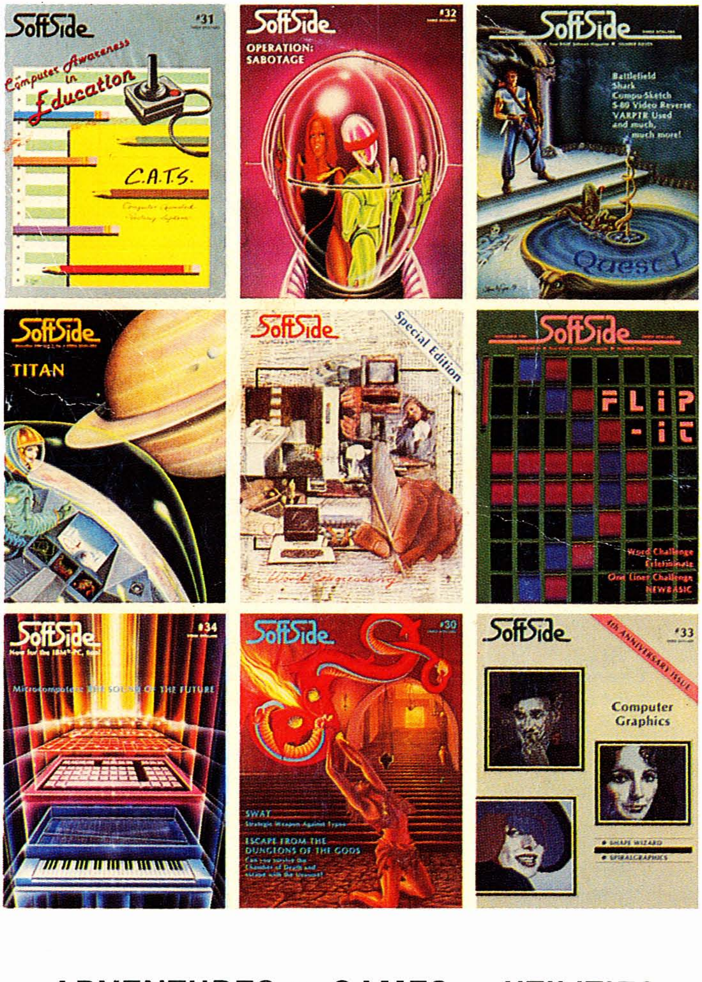
ATARI Edition

THE BEST OF SoftSide™



ADVENTURES • GAMES • UTILITIES

THE BEST OF SoftSide™

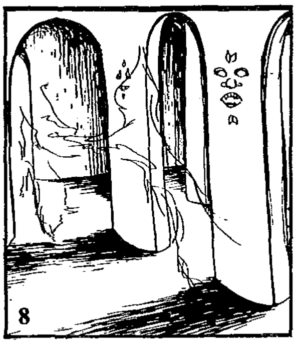
ATARI® PROGRAMS for the Atari® Personal Microcomputers

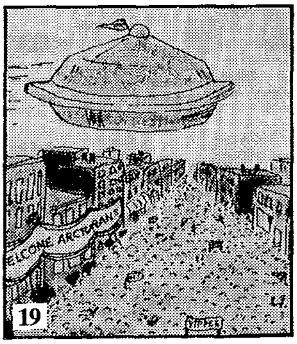
Edited by Fred Condo

Copyright © 1983
SOFTSIDE PUBLICATIONS, INC.
Milford, New Hampshire

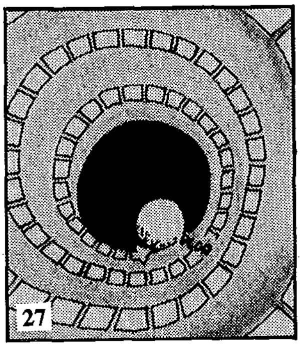
THE BEST OF SoftSide™

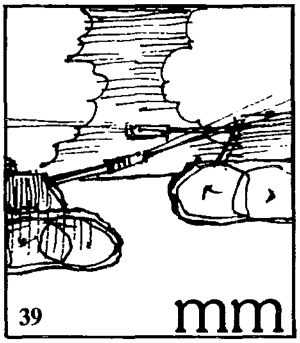
CONTENTS

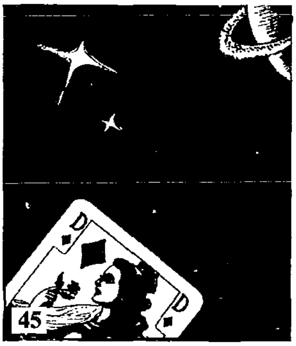

8



19


27


39
mm


45

51

61

INTRODUCTION

Over four years ago, when the microcomputer revolution was little more than a low rumble, ***SoftSide***, "Your BASIC Software Magazine," published its first issue. In its early days, ***SoftSide*** supported only one machine, the now-venerable TRS-80® Model I. As the revolution gained momentum in the last years of the 1970's, ***SoftSide*** instituted a separate version of the magazine for the new Apple II® personal computer. The Apple added something novel to the microcomputerist's world — color! What could come next?

In August of 1980, ***SoftSide*** initiated a new format. That issue was the first combined magazine for the TRS-80 and Apple with the addition of another new personal computer, the Atari® 400/800. ***SoftSide*** was now providing quality BASIC software for three computers at a price that made it a great bargain (and still does).

SoftSide has always relied on program submissions from its readers. Few publications have a readership so intimately involved with its month-to-month features. As ***SoftSide*** grew and improved, so did you, our readers. You made BASIC do previously unimaginable things. Through ***SoftSide***, the once-mute TRS-80 and Apple computers were given voices, and the Atari veritably sang.

Though its main purpose has always been to provide the best entertainment software in BASIC for the home computer user, ***SoftSide*** has explored every facet of personal computing. The book you now hold contains utilities, space games, adventures, an information manager, a text editor, and more — in all, over twenty fully updated programs to amuse you and help you use your personal computer to greater advantage. Included are some of our most popular programs ever, *Quest, Flip-It,* and *SWAT.* Most of the programs have been fully documented by the authors, and are accompanied by annotated variable lists, to help you learn more about BASIC and the way the programs accomplish what they do. If you are a regular reader of ***SoftSide***, these are familiar features. If you're new to us, welcome to ***The Best of SoftSide.***

USING THE BEST OF SOFTSIDE

You are now holding the Atari version of ***The Best of SoftSide.*** Also available are the Apple and TRS-80 versions.

This book contains over twenty programs and their accompanying explanatory articles and documentation selected from past issues of ***SoftSide*** magazine. Each program is introduced and explained by the article, followed by a listing of the BASIC program itself. These listings were generated by an Atari computer just like yours, and appear in a 38-column format, the same as the Atari's screen display.

One of the features of the Atari computer is its multiplicity of special graphics characters and inverse video. The graphics characters are usually control characters — that is, you type them by pressing a key while holding down the key marked CTRL. Inverse video is toggled on and off by the Atari logo key (⋀). These special characters are somewhat difficult to represent in a printed listing, so we have adopted some simple conventions to ease your typing. See the next page for examples and explanations of these conventions.

The greatest aid to your typing is *SWAT*, the Strategic Weapon Against Typos. This debugging program appears last in the book, and is followed by "*SWAT* Tables" for every program. The *SWAT* article will explain how to use the program to locate and elimiate typographical errors. Many of the programs in ***The Best of SoftSide*** originally appeared before the development of *SWAT*, and appear here with *SWAT* Tables for the first time.

If typing doesn't sound like fun to you, be sure to see the special order form for the disk version of ***The Best of SoftSide*** elsewhere in this book.

And now, we proudly present ***The Best of SoftSide.***

PRINTED LINE LISTING CONVENTIONS

Listed below is a short example of an Atari program listed according to ***SoftSide's*** standard conventions. The program, of course, is trivial, and doesn't even run; it is just a short example that should make our listing methods clear.

```
 10 PRINT "}"
20 A$="mmmmmffffgalllmmmmm":REM Lower
case characters are control.
30 G7$="ENEMY: ard":REM Lower case cha
racters are control.
40 R$="*=-+":REM Control characters.
```

In line 10, you see the right brace character, " } ". This represents the clear-screen character, whose ATASCII code is 125. You type this character by first pressing and releasing the ESC key, located in the upper left quadrant of your keyboard, then pressing SHIFT-CLEAR. On your Atari's screen, this will appear as an arrow that curves up and to the left. Go ahead and try this example on your computer.

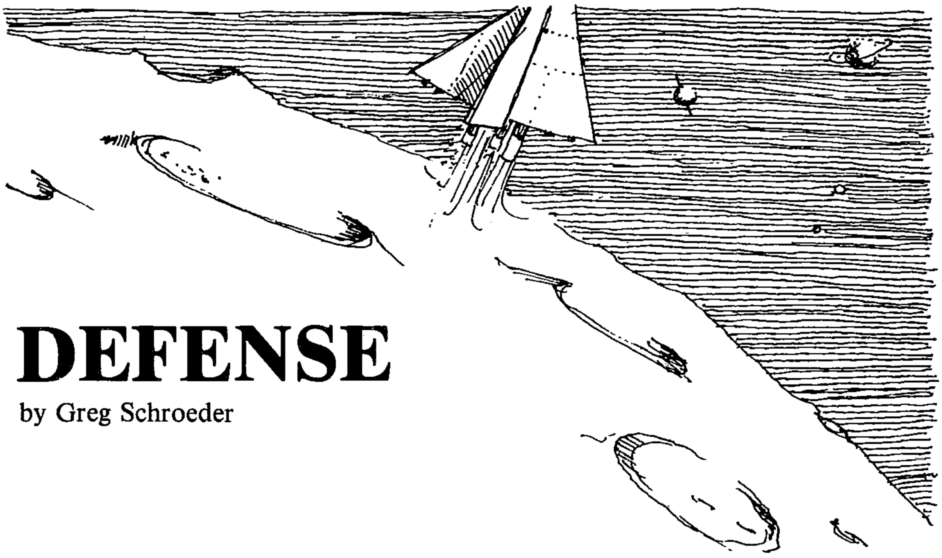
Let's go on to line 20. Notice the REM statement. It will be appended to the line in question, or it will occupy the immediately preceding line. Those lower-case characters represent the unprintable control (graphics) characters that your Atari can display. These are typed by pressing the appropriate letter key *while* you hold down the CTRL key. The string in line 20 begins with 5 CTRL-M's. Try it!

In line 30, you can see the REM statement warning you about control characters. Also note the underlining. This represents characters to be typed in inverse video. You put your Atari in inverse video by pressing and releasing the Atari logo key at the right end of the space bar. Pressing it again puts the computer back in the normal video mode. To type the string in line 30, you would press the Atari logo key before typing the first "E". Then you would type "ENEMY: ". Note that the space after the colon *is* underlined; you must type it in inverse, as well. Now you've come to the lower-case letters. The REM tells you that these are graphics (control) characters. The "a" isn't underlined, so you should type the CTRL-A that it represents in *normal* video. The CTRL-R and CTRL-D are in inverse. Yes, even graphics characters can be either normal or inverse. The proper typing of inverse is important both to getting a correct display on the screen and to generating proper *SWAT* tables.

Finally, we come to line 40. The string here has cursor-control characters in it — the arrow keys. Normally, when you press CTRL and an arrow key, the cursor moves in the direction of the arrow. These movement codes can be put into programs, too, but they don't show up on a printer. Since the arrow keys have other characters on them, we can use them to represent the arrow keys, with an accompanying REM to warn you.

You type these characters by pressing and releasing ESC, then pressing the appropriate arrow key while holding down CTRL.

That's really all there is to it, so go ahead and delve into ***The Best of SoftSide.***



DEFENSE

by Greg Schroeder

***Defense* is an arcade-style game for one or two players requiring an Atari with 16K RAM (32K with disk) and one joystick.**

The object of *Defense* is to destroy as many of the aliens that fly across the moon's surface as possible, before your three fighter ships have been destroyed.

When the game begins, you are given a brief description of your situation and mission while the computer redefines the Atari character set. Then you type 1 or 2 for the number of players in the game. A "get ready" prompt is given at the start of each player's turn. The computer will draw a text graphics display using the characters it previously defined in graphics mode 1, showing at the top each player's score, the highest score so far, and number of ships left; and at the bottom, how many aliens in the current attack wave remain. In the center is your playing field, with your special text graphics ship on the left side and the moon's surface along the bottom.

Player #1's joystick is plugged into socket #2, and player #2's joystick plugged into socket #3. If you push the joystick up or down, your ship will be raised or lowered. But don't go too far down, because if you hit the ground, your ship will be destroyed. Push the joystick to the right and after about a second, you will gain airspeed and the ground will start to move. Pushing the stick to the left will cause you to lose airspeed and the ground to slow also.

As many as three aliens at a time will appear on the screen when they have reached your section of the moon's surface. The scoring system for each alien is as follows:

Yellow Drone: 10 points.
Green Drone: 20 points.
Blue Drone: 30 points.
Red Drone: 40 points.
Red Smiling Blob: 100 points.
Green Smiling Blob: 200 points.

To destroy these aliens, you must reach their altitude with your ship and fire a laser beam by pressing the fire button. When you hit them, their scores are shown and added to your total score.

The aliens fly in specific attack patterns. The drones will fly diagonally, bouncing off the top and bottom of the playing field, while the smiling blobs fly in an erratic pattern towards you and have a better chance of hitting you. Be careful, because the smiling blobs begin in the center of the screen, unlike the drones which start at the far right.

If an alien does hit you, your ship will be shown exploding in a graphics display and one ship will be deducted from your possession. Then, if it is a one-player game, you will continue in a different ship, or if a two-player game, the next player will have a turn.

When you run out of ships the game is over for you and your score is displayed. But there is a way to gain extra ships. As you destroy the aliens, the number of remaining attackers in that wave will decrease on the bottom of the screen. If all the aliens in that wave are destroyed, then you will be given 500 bonus points for each ship you have left. Every time you clear out four such waves, you are awarded a bonus ship.

Note: The system reset key must be pressed before each run of the program to restore the Atari character set.

Variables

A(n,n): Nonvisible position of aliens along the moon's surface.
A, I, J, X,: Miscellaneous variables.
AL(n): Number of aliens in each player's current attack wave.
A$: Moon surface picture string.
B(n), C(n): X and Y positions of aliens on screen.
D(n): Y movement of alien on screen.
E(n): Type of alien on screen.
F(1), F(2): Number of aliens in each player's attack wave. Used to reset AL(n).
F(3), F(4): Number of attack waves destroyed by each player.
GS: Ground speed. Controls speed of ground picture movement and alien movement.
H$: Temporary string storage used in moving the moon's surface.
PL: Current player number.
PP: Total number of players.
R(n): Stores number of aliens along moon's surface that are on the screen.
S: Joystick value.
SC(n): Each player's score.
SH(n): Remaining ships for each player.
SL: Trigger flag. If SL = 1 then the trigger is still pressed.
ST: Pointer to beginning of redefined character set.
TT: Counter for alien movement on screen.
XC: Counter for alien movement along moon's surface.
Y: Y axis of postion of player's ship. (X defaults to 1.)
Y1: Temporary storage for Y.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS          'Defense'          SS
SS   Author:  Greg Schroeder   SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Lines 10-50: Initialization

Jump to subroutine for title and character redefinition.

```
10 GOSUB 10000
```

Dimension variables.

```
15 DIM A(2,41),B(4),C(4),D(4),E(4),F(4
),SH(2),SC(2),A$(21),H$(5),R(4),AL(2):
SC(1)=0:SC(2)=0
```

Set arrays to zero.

```
20 FOR I=0 TO 4:B(I)=-1:C(I)=0:D(I)=0:
E(I)=0:R(I)=0:NEXT I
25 I=SC(1)*(SC(1)>SC(2))+SC(2)*(SC(2)>
SC(1)):HSC=HSC*(HSC>I)+I*(I>HSC)
```

Place aliens for two players in moon's surface variable.

```
30 FOR I=1 TO 2:FOR J=1 TO 20:A(I,J)=I
NT(RND(0)*16):NEXT J:NEXT I
```

Define other variables for both players.

```
40 SH(1)=3:SH(2)=3:SC(1)=0:SC(2)=0:AL(
1)=20:AL(2)=20:F(1)=20:F(2)=20:F(3)=0:
F(4)=0:XC=0
```

Jump to "get ready" subroutine.

```
50 GOSUB 6000
```

Lines 100-150: Main loop.
Increment alien on screen counter.

```
100 TT=TT+1:IF TT=4 THEN TT=1
```

Move moon's surface contained in A$.
Use H$ as temporary storage.

```
105 IF GS>1 THEN H$=A$(1,GS):A$=A$(GS+
1,20):A$(LEN(A$)+1)=H$:POSITION 0,20:?
 #6;A$;
```

Lines 110-140: Move ship.
Check if joystick moved.

```
110 S=STICK(PL):IF S=15 THEN 150
```

Check joystick to move ship up.

```
120 Y1=Y:IF S/2=INT(S/2) AND Y>6 THEN
Y=Y-1
```

Check joystick to move ship down. See if ship hit ground.

```
125 IF (S=5 OR S=9 OR S=13) THEN Y=Y+1
:IF Y=20 THEN GOTO 3000
```

Erase and reprint redefined ship character.

```
130 POSITION 1,Y1:? #6;"  ";:POSITION
1,Y:? #6;"XY";
```

Check joystick to move ship faster.

```
135 IF S>4 AND S<8 AND GS<2 THEN GS=GS
+0.05
```

Check joystick to move ship more slowly.

```
140 IF S>8 AND S<12 THEN GS=GS-0.05:IF
 GS<0 THEN GS=0
```

Lines 150-170: Fire laser.
Check if trigger released.

```
150 IF STRIG(PL)=1 THEN SL=0:GOTO 200
```

If trigger is still pressed, branch to alien movement.

```
155 IF SL=1 THEN 200
```

Print redefined text graphics laser. Make noise. Check if laser hit alien(s).

```
160 SOUND 1,20,4,15:POSITION 3,Y:? #6;
"................";:FOR I=1 TO 3:IF B(
I)>2 AND INT(C(I))=Y THEN 4000
161 REM LINE 160: 16 INVERSE <CTRL>'.'
162 NEXT I
```

Erase laser. Turn off sound. Set trigger pressed variable.

```
165 POSITION 3,Y:? #6;"
  ";:SOUND 1,0,0,0:SOUND 2,0,0,0:SL=1
```

Check if all aliens are gone.

```
170 IF AL(PL)<1 THEN 4100
```

Lines 200-300: Alien movement.
Increment moon surface counter.

```
200 XC=XC+1:IF XC=F(PL)+1 THEN XC=1
```

Move aliens' position in array.

```
210 IF A(PL,XC)>-5 THEN A(PL,XC)=A(PL,
XC)-(GS):IF A(PL,XC)<=0 THEN 1000
```

Check if array contains no aliens.

```
220 IF B(TT)=-1 THEN 300
```

Erase alien. Move alien along X axis.
Check if it is off-screen.

```
230 POSITION INT(B(TT)),INT(C(TT)):? #
6;" ";:B(TT)=B(TT)-GS-0.5:IF B(TT)<0 T
HEN B(TT)=-1:A(PL,R(TT))=15:GOTO 300
```

Check if alien is smiling blob, and
branch to different movement routine.

```
240 IF E(TT)>4 THEN 2000
```

Check if alien is off-screen after moving
along Y axis.

```
250 C(TT)=C(TT)+D(TT):IF C(TT)<6 OR C(
TT)>19 THEN C(TT)=C(TT)-D(TT):D(TT)=-D
(TT)
```

Check if alien hit player.

```
255 IF INT(B(TT))>=0 AND INT(B(TT))<=2
 THEN IF INT(C(TT))=Y THEN 3000
```

Print specific redefined alien character.

```
260 POSITION INT(B(TT)),INT(C(TT)):GOS
UB 260+E(TT)*5:GOTO 300
265 ? #6;"Z";:RETURN
270 ? #6;"z";:RETURN
275 ? #6;"Z";:RETURN
280 ? #6;"z";:RETURN
285 ? #6;"t";:RETURN
290 ? #6;"t";:RETURN
```

Return to start of main loop

```
300 GOTO 100
```

Lines 1000-2010: Set up new alien.
Find empty alien variable. If none, then
restart alien at end of moon's surface.

```
1000 FOR I=1 TO 3:IF B(I)=-1 THEN 1010
1005 NEXT I:A(PL,XC)=15:GOTO 220
```

Set Y value for alien. Set alien's
movement.

```
1010 C(I)=INT(RND(0)*12)+7:D(I)=RND(0)
+0.1-1.5*(RND(0)<0.5)
```

Pick alien type. Remember which
surface alien it is. If alien is a smiling
blob, then start X position at mid-screen.

```
1020 E(I)=INT(RND(0)*6)+1:R(I)=XC:A(PL
,XC)=-10:B(I)=17:IF E(I)>4 THEN B(I)=1
1
1025 GOTO 220
```

Move smiling blob along Y axis.

```
2000 IF C(TT)<Y THEN D(TT)=1:GOTO 250
2010 IF C(TT)>=Y THEN D(TT)=-1:GOTO 25
0
```

Lines 3000-3070: You've been hit.
Clear playing field.

```
3000 FOR I=6 TO 19:POSITION 0,I:? #6;"
                ";:NEXT I
```

Reset aliens to surface start.

```
3010 FOR I=1 TO 3:IF B(I)>0 THEN A(PL,
R(I))=15:B(I)=-1
```

Turn off sound.

```
3015 NEXT I:FOR I=0 TO 3:SOUND I,0,0,0
:NEXT I
```

Print redefined ship character. Brighten
ship in red.

```
3020 POSITION 1,Y:? #6;"XY";:FOR I=2 T
O 8 STEP 0.1:SETCOLOR 2,4,I:NEXT I
```

Make explosion sound, first explosion. Initialize explosion loop.

```
3025 SOUND 0,20,8,15:POSITION 1,Y:? #6
;"**";:FOR I=15 TO 0 STEP -0.2
```

Make explosion sounds.

```
3030 SOUND 0,20,8,I:SOUND 1,75,8,I+1:S
OUND 2,175,8,I+3:SOUND 3,255,8,I+5
```

Print explosion steps when ready.

```
3040 IF I=12 THEN POSITION 0,Y-1:? #6;
" **";:POSITION 0,Y:? #6;"*::*";:POSIT
ION 1,Y+1:? #6;"**";
3045 IF I=8 THEN POSITION 0,Y-1:? #6;"
*.:*";:POSITION 0,Y:? #6;"%++#";:POSIT
ION 0,Y+1:? #6;"*:.*";
3050 IF I=4 THEN POSITION 0,Y-1:? #6;"
 + .";:POSITION 0,Y:? #6;"+..+";:POSIT
ION 1,Y+1:? #6;"',";
3055 IF I=1 THEN POSITION 0,Y-1:? #6;"
    ";:POSITION 0,Y:? #6;"    ";:POSIT
ION 0,Y+1:? #6;"    ";
```

Turn off sound. Erase ship.

```
3060 NEXT I:FOR I=0 TO 3:SOUND I,0,0,0
:NEXT I:POSITION 1,Y:? #6;"  ";
```

Set ground speed to zero. Subtract ship from player. Print number of ships left. Check if all ships gone.

```
3062 GS=0:SH(PL)=SH(PL)-1:POSITION 3+1
2*(PL=2),4:? #6;SH(PL);:IF SH(PL)=0 TH
EN GOSUB 7000
```

If there is another player, then change player variable.

```
3065 IF SH(PL-1+2*(PL=1))>0 THEN PL=PL
+1:IF PL>PP THEN PL=1
```

Jump to "get ready" routine. Jump to screen drawing.

```
3070 GOSUB 5000:FOR I=1 TO 1000:NEXT I
:GOSUB 6000:XC=0:GOTO 100
```

Destroy aliens. Make noise. Check if alien is a smiling blob.

```
4000 SOUND 2,RND(0)*200+50,10,10:POSIT
ION INT(B(I)),INT(C(I)):? #6;:IF E(I)>
4 THEN 4020
```

Print score for drone. Add to total score.

```
4010 ? #6;E(I)*10;:SC(PL)=SC(PL)+E(I)*
10:GOTO 4030
```

Print score for smiling blob. Add to total score.

```
4020 ? #6;(E(I)-4)*100;:SC(PL)=SC(PL)+
(E(I)-4)*100
```

Clear alien variables. Print player's score.

```
4030 B(I)=-1:A(PL,R(I))=-5:POSITION 2,
3:? #6;SC(1);:IF PP=2 THEN POSITION 14
,3:? #6;SC(2);
```

Decrease the number of aliens in attack wave. Return to see if more hit.

```
4040 AL(PL)=AL(PL)-1:POSITION 2,22:? #
6;"zt=";AL(1);" ";:IF PP=2 THEN POSITI
ON 12,22:? #6;"Zt=";AL(2);" ";
4050 NEXT I:GOTO 165
```

Lines 4100-4135:

Attack wave destroyed. Clear screen. Restore Atari character set. Add to player's wave cleared total. Print message. Make noise.

```
4100 GRAPHICS 17:POKE 756,224:F(PL+2)=
F(PL+2)+1:POSITION 8,10:? #6;"attack":
? #6;"  wave ";F(PL+2);
4110 ? #6;" destroyed":? #6:SOUND 0,20
0,10,10:SOUND 1,201,10,10:SOUND 2,0,0,
0
```

Print bonus message (500 points for each remaining ship). Add bonus to total score.

```
4120 ? #6;"    BONUS - ";SH(PL)*500:SC
(PL)=SC(PL)+SH(PL)*500
```

Check if extra ship is awarded. Print message.

```
4123 IF F(PL+2)/4=INT(F(PL+2)/4) THEN
SH(PL)=SH(PL)+1:? #6:? #6;"    extra s
hip"
```

Add more to next wave. Reset aliens on surface.

```
4125 AL(PL)=F(PL)+5:F(PL)=F(PL)+5:IF F
(PL)>40 THEN F(PL)=40:AL(PL)=40
```

Make sure wave never exceeds 40 aliens.

```
4130 FOR J=1 TO AL(PL):A(PL,J)=INT(RND
(0)*16):NEXT J
```

Jump to screen drawing. Jump to main loop.

```
4135 FOR I=1 TO 1500:NEXT I:GOSUB 6000
:GOTO 100
```

"Get Ready" Clear screen. Restore Atari character set. Print message.

```
5000 GRAPHICS 18:POKE 756,224:POSITION
 4,5:? #6;"PLAYER   ";PL
5005 POSITION 4,6:? #6;"GET READY!!":R
ETURN
```

Lines 6000-6070:

Draw screen. Clear screen. Restore new character set. Turn off sound.

```
6000 GRAPHICS 17:POKE 756,ST/256:FOR X
=0 TO 3:SOUND X,0,0,0:NEXT X
```

Print message for both players (if only one player, print only his messages).

```
6020 POSITION 6,0:? #6;"defense":POSIT
ION 3,2:? #6;"#1 ^^^^^^^  #2":POSITION
 (18-LEN(STR$(HSC)))/2,1:? #6;HSC;
6030 POSITION 2,3:? #6;SC(1);:POSITION
 3,4:? #6;SH(1);
6040 IF PP=2 THEN POSITION 14,3:? #6;S
C(2);:POSITION 15,4:? #6;SH(2);
6045 POSITION 2,22:? #6;"zt=";AL(1);:I
F PP=2 THEN POSITION 12,22:? #6;"Zt=";
AL(2);
```

Print line made of redefined characters.

```
6050 FOR I=0 TO 19:POSITION I,5:? #6;"
W";:NEXT I
```

Assemble moon's surface picture in A$.

```
6060 FOR I=1 TO 20:A$(I,I)="@":NEXT I:
A$(10,11)="UV":A$(2,3)="UV":A$(19,20)=
"UV"
```

Print ground. Set ship position (Y) to mid-screen, and print ship. Make rocket sound.

```
6070 POSITION 0,20:? #6;A$;:Y=12:POSIT
ION 1,Y:? #6;"XY";:SOUND 0,75,8,4:RETU
RN
```

Lines 7000-7050:

Game over. Restore Atari character set. Print game over message and score.

```
7000 POKE 756,224:GRAPHICS 18:POSITION
 5,5:? #6;"PLAYER ";CHR$(PL+176);:POSI
TION 5,6:? #6;"game over";
7005 POSITION 4,8:? #6;"score = ";SC(P
L);
```

See if both players' games are over.

```
7010 FOR I=1 TO 400:NEXT I:IF SH(1)=0
AND SH(PP)=0 THEN 7050
```

Return to main loop.

```
7020 RETURN
```

POP last GOSUB from stack. Jump to title routine. Jump to number of players input. Jump to set-up of variables.

```
7050 FOR I=1 TO 400:NEXT I:POP :GOSUB
```

Lines 10000-10070:

Redefine character set. Set top of memory five pages down. Initialize graphics. Jump to title display routine.

```
11000:GOSUB 11100:GOTO 20
10000 POKE 106,PEEK(106)-5:GRAPHICS 0:
GOSUB 11000
```

Set start of new character set pointer.

```
10010 ST=(PEEK(106)+1)*256
```

Move Atari character set from ROM to top of RAM. Jump to number of players input.

```
10020 FOR X=0 TO 1023:POKE ST+X,PEEK(5
7344+X):NEXT X:GOSUB 11100
```

Redefine "X" character to be back half of ship. Redefine "Y" to be front half. "Z" becomes a drone. "U" is CHR$(6). "V" is CHR$(7).

```
10030 FOR X=0 TO 23:READ A:POKE ST+448
+X,A:NEXT X:FOR X=0 TO 15:POKE ST+424+
X,PEEK(ST+560+X):NEXT X
```

"W" is CHR$(18)

```
10040 FOR X=0 TO 7:POKE ST+440+X,PEEK(
ST+656+X):NEXT X
```

"T" is a smiling blob.

```
10045 FOR X=0 TO 7:READ A:POKE ST+416+
X,A:NEXT X
```

"@" is flat ground and laser.

```
10050 FOR X=0 TO 6:POKE ST+256+X,0:NEX
T X:POKE ST+263,255
```

Initialize new character set.

```
10055 GRAPHICS 1:POKE 756,ST/256:RETUR
N
```

DATA for redefined characters.

```
10060 DATA 0,0,0,0,30,15,7,7,0,0,0,0,0
,192,252,255,0,8,28,62,127,42,20,34
10070 DATA 0,126,153,153,255,189,195,1
26
```

Title display. Suppress cursor. Print title and instructions.

```
11000 GRAPHICS 17:POKE 752,1:POSITION
7,0:? #6;"defense":? #6:? #6:? #6;"  e
vil aliens from"
11010 ? #6;" beyond earth have   invad
ed the moon.":? #6:? #6
11020 ? #6;"  YOUR MISSION IS    TO DE
STROY AS MANY  ATTACKING WAVES OF  THO
SE ALIENS BEFORE";
11030 ? #6;" THEY LAUNCH THEIR   FLEET
S TO CONQUER   THE EARTH.":? #6:? #6
11040 ? #6;"  YOUR JOYSTICK      CONTR
OLS YOUR SPEED AND ALTITUDE. PUSH  THE
 BUTTON TO FIRE."
11050 RETURN
```

Lines 11100-11120:

Number of players input. OPEN keyboard. Print message.

```
11100 OPEN #2,4,0,"K:":POSITION 0,23:?
 #6;"PUSH 1 OR 2 TO START";
```

Check if key "1" or "2" was pressed.

```
11110 IF PEEK(764)<>30 AND PEEK(764)<>
31 THEN 11110
```

GET key pressed. CLOSE keyboard. Decide whether one or two people are playing.

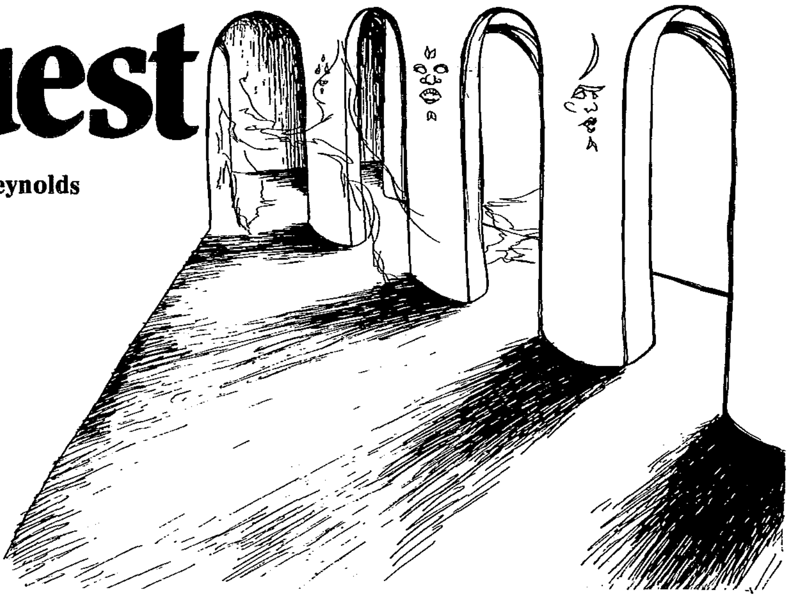
```
11115 GET #2,I:PP=1:CLOSE #2:IF I=50 T
HEN PP=2
```

Set player variable. Jump to "get ready" subroutine.

```
11120 PL=1:GOSUB 5000:RETURN
```

Quest

by Brian Reynolds



Atari version by Alan J. Zett

"Quest 1" is a graphic adventure game that runs in 24K RAM.

In "Quest 1" you become a strong warrior who journeys through an ancient maze in search of four huge sapphires and other treasures. These precious jewels are guarded by terrible Wraiths, Giants, Mummies, and other unpleasant monsters. To find the treasures, then, you must be very strong (to kill the monster) or very dextrous (to sneak around the monster and steal the treasure).

When you begin your quest, a character will be created for you. He (or she) will be either an Elf, a Dwarf, or a Human. He will be given ratings in strength (3-20) and dexterity (3-20), and a percentage rating according to his wounds (100% = no wounds, 0% = demise). Being new to the field of questing, your warrior will not be much favored by the gods and will not have much magic to use. He will, however, have four different ways to fight: He will be given a random number of normal arrows, magic arrows, and holy water, plus his trusty sword. Some healing potions will also be given for restoring wounds.

After you have named your fighter, you will be teleported into a marketplace in a nearby town to bargain with a greedy merchant for more supplies. This usually takes only a short time, since the merchant will probably either sell to you quickly, or else refuse to sell at any kind of affordable price. After completing your bargaining, enter a '0' to begin your quest.

When you enter the dungeon, a text-graphic display will be created on the screen, showing all your statistics in the corners and a picture of your current location in the center. A specially-defined text character represents you. Treasure chests appear as asterisks (*), while monsters are shown by the initial letter of their name.

You can attempt your quest through the 58 rooms of the dungeon simply by killing monsters, taking the treasures, and moving on. However, this is not advisable for two reasons. First, you must remember the way out of the dungeon, or you will surely perish. And second, wandering monsters abound in this dungeon; if a wraith, for example, comes up behind you, he will probably kill you with one good blow. You should also be aware that frequent trips back for supplies are not wise, since, once you've garnered 100 experience points, more monsters enter the dungeon while you're shopping.

Note that the greater your dexterity rating, the more *slowly* the game will seem to move. This is because your higher dexterity, in effect, gives you more time to think and react relative to the pace of the game. As you accumulate experience points, however, the pace and difficulty of the game will increase.

When you find your way out of the dungeon, the computer will give you a list of all the treasures you retrieved, add in any arrows or potions you may have found, give you a chance to save the game, and let you quit if you want to. If you do quit, the computer will give you a list of all your fighter's abilities and possessions so that you can use him in a later game. If you elect to continue, you are teleported back into the marketplace to get more supplies and then to continue your quest.

Commands are entered with single keystrokes, as follows:

W: Move up.
A: Move left.
D: Move right.
X: Move down.

Any key other than above: Stop movement.
N: Shoot a normal arrow (not effective against Wraiths).
M: Shoot a magic arrow.
T: Toss a vial of holy water (affects only "undead" monsters: Skeletons, Zombies, Ghouls, Mummies).
F: Fight in close combat (not effective against Giants or Wraiths).
O: Open a treasure chest when you are next to it. (It will disappear and its contents will be displayed on the screen.)
H: Drink a healing potion. (This restores your wound rating to 100%.)

On page 18 is a listing of *Quest 2 Data*. You should type this listing as a *separate* program, and check it with *SWAT*. Then save it on tape or disk with the LIST "C:" or LIST "D: filename" command.

Quest 2 Data supplies the information for a different dungeon. If you get tired of *Quest 1's* original dungeon, you can change dungeons by first loading *Quest 1*, then issuing the ENTER "C:" or ENTER "D:file name" command, as appropriate. Then save the resultant new program under a different filename or on another tape. Have fun!

Variables

A1: Number of normal arrows.
A2: Number of magic arrows.
DX: Dexterity rating.
EP(*): Experience value of each treasure.
GP(*): Gold value of each treasure.
HW: Number of vials of holy water.
M$(*): Single-character monster identifier.
M1(*): Type of monster in each room.
M2(*): Number of monsters in each room.
MN$(*): Names of the monsters.
MS(*): Standard wound value for each monster.
NM$: Name of fighter character.
OP: Original price of an item at the market.
P1: Current price of an item at the market.
PT: Number of healing potions.
R1(*): Identifies each location as either a passage/intersection (=1) or a chamber/room (=2).

R2(*,*): For each room, identifies what room you will enter by exiting up, down, left, and right respectively.
RC: Race of fighter (0 = Human, 1 = Elf, 2 = Dwarf).
RM: Current room number.
ST: Player strength.
T$(*): Name of each treasure.
T1(*): Identifies type of treasure in each room.
TS(*): Quantity of each treasure type retrieved by player.
TX,TY: X and Y coordinates of treasure.
W: Wounds (multiply by 100 to get percentage).
WX,WY: X and Y coordinates of monster.
X5,Y5: X and Y coordinates of player.
YY$: Single-character identifier for player.

Editor's notes:

XFR$, which you will find printed as a series of periods in the program listing, actually contains a Machine Language routine that quickly sets up a redefined character set for the Atari. It is very difficult to represent this string in the listing of the program, so here are explicit instructions on how to type it.

In these instructions, each key is represented by something between brackets. Thus [h] means to type a lower-case "h"; likewise, [Atari] means to press the Atari logo key, which is found next to the right-hand shift key, and [CTRL-N] means to press the "N" key while holding down the "CTRL" key. Special note: [1] is a the numeral one — it is not the lower-case version of "L"!

Type the following sequence exactly to produce XFR$:
[h][Atari][)][Atari] [CTRL-,] [Atari] [CTRL-E] [K] [CTRL-E] [M] [)] [CTRL-.] [CTRL-E] [N] [%] [Atari] [j] [CTRL-X] [i] [CTRL-A] [Atari] [CTRL-E] [L] [spacebar] [Atari] [CTRL-,] [Atari] [1] [M] [CTRL-Q] [K] [H] [P] [y] [f] [L] [f] [N] [%] [N] [l] [d] [P] [m] [Atari] [CTRL-.]

User must hit [System Reset] prior to every "RUN" of "Quest" to reinitialize the character set.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS           'Quest'          SS
SS  Author: Brian Reynolds    SS
SS Translator: Alan J. Zett   SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

Initialize character modification and print title page.

```
1 GOSUB 30000:? CHR$(125):POKE 752,1:P
OSITION 15,6:? " QUEST I "
2 ? :? "QUEST was written by Brian Rey
nolds":OPEN #3,4,0,"K"
3 ? :? "(ATARI translation by Alan J.
Zett)"
7 GOSUB 600:FOR X=1 TO 276:T$(X)=" ":N
EXT X:FOR X=1 TO 88:MN$(X)=" ":NEXT X:
YY$="@"
8 FOR X=0 TO 9:TS(X)=0:NEXT X
9 FOR X=1 TO 49:K$(X,X)=CHR$(INT(RND(0
)*32)):NEXT X:K$(49)=" "
```

Data for monsters and treasures.

```
100 DATA Worthless odds and ends,0,0,A
 bag of Copper Coins,1,3,A small Brass
 Statuette,2,5
102 DATA A bag of various Coins,3,7,A
purse of Gold Coins,5,12,3 Gold Nugget
s,8,17
104 DATA 4 small Turquoises,7,15,A lar
ge Ruby,15,30
106 DATA A *HUGE* Sapphire,150,150,A H
```

```
ealing Potion,10,0
108 DATA 10 Magic Arrows,15,0,10 norma
l Arrows,10,0
110 DATA SKELETON,S,2,ORC,O,3,ZOMBIE,Z
,4,GHOUL,g,6
112 DATA HUGE SPIDER,H,7,MUMMY,M,8,GIA
NT,G,9,WRAITH,W,9.9
```

Data for the rooms.

```
115 DATA 1,12,3,2,18,0,0,0
120 DATA 2,0,0,0,1,4,2,8
125 DATA 1,1,0,4,19,0,0,1
130 DATA 1,0,0,5,3,3,1,1
135 DATA 2,6,38,0,4,1,3,6
140 DATA 1,8,5,9,7,0,0,0
145 DATA 1,0,0,6,0,0,0,1
150 DATA 2,0,6,0,11,2,11,2
155 DATA 2,0,0,10,6,2,3,1
160 DATA 2,0,0,0,9,5,1,4
165 DATA 1,0,0,8,12,0,0,1
170 DATA 2,0,1,11,13,2,5,3
175 DATA 1,0,0,12,14,0,0,1
180 DATA 2,15,26,13,17,5,1,1
185 DATA 2,0,14,0,0,0,0,1
190 DATA 2,0,17,0,0,1,2,5
195 DATA 1,16,20,14,0,4,1,1
200 DATA 2,0,19,1,26,2,2,7
205 DATA 2,18,30,3,27,3,2,2
210 DATA 1,17,21,0,0,0,0,1
215 DATA 1,20,22,0,0,6,2,9
220 DATA 1,21,23,0,0,2,3,12
225 DATA 1,22,24,0,0,4,2,10
240 DATA 1,23,25,34,0,0,0,11
250 DATA 2,24,0,0,0,7,3,9
260 DATA 2,14,0,18,0,3,2,1
270 DATA 2,0,28,19,0,4,1,2
280 DATA 1,27,29,31,0,0,0,1
290 DATA 2,28,0,0,0,5,1,10
300 DATA 2,19,0,0,0,1,2,3
310 DATA 1,0,32,0,28,0,0,4
320 DATA 1,31,33,43,0,0,0,1
330 DATA 2,32,35,0,0,5,1,8
340 DATA 1,0,0,35,24,0,0,12
350 DATA 1,33,36,45,34,0,0,5
360 DATA 1,35,0,37,0,7,1,10
370 DATA 2,0,0,0,36,8,3,9
380 DATA 1,5,49,0,39,0,0,1
390 DATA 1,0,40,38,0,0,0,6
400 DATA 1,39,0,0,41,2,3,2
410 DATA 1,42,46,40,43,4,1,7
420 DATA 2,0,41,0,0,7,3,8
430 DATA 2,0,44,41,32,6,1,11
440 DATA 1,43,45,0,0,0,0,5
450 DATA 1,44,0,47,35,0,0,1
460 DATA 2,41,47,48,0,5,1,7
470 DATA 1,46,0,50,45,0,0,3
480 DATA 1,0,0,49,46,0,0,1
490 DATA 2,38,51,52,48,6,1,6
500 DATA 1,0,0,51,47,2,5,10
510 DATA 1,49,0,53,50,4,3,5
520 DATA 2,0,0,0,49,6,1,6
530 DATA 2,0,54,0,51,5,1,8
540 DATA 1,53,0,0,55,0,0,1
550 DATA 2,0,0,54,56,2,3,2
560 DATA 1,0,0,55,57,6,1,8
570 DATA 1,0,0,56,58,7,3,11
580 DATA 2,0,0,57,0,8,4,9
590 RESTORE :GOTO 603
```

Initialize the variables.

```
600 DIM MN$(89),M$(8),MS(8),R1(58),R2(
58,4),M1(58),M2(58),T1(58),T$(277),EP(
12),GP(12),TS(9),YY$(1)
602 DIM X$(23),NM$(20),A$(5),K$(49),X1
$(2),I$(2):RETURN
603 FOR X=1 TO 12:READ X$,X1,X2:T$((X*
23)-22)=X$:EP(X)=X1:GP(X)=X2:NEXT X:T$
(277)="#"
605 FOR X=1 TO 8:READ X$,X1$,X2:MN$((X
*11)-10)=X$:M$(X)=X1$:MS(X)=X2:NEXT X:
MN$(89)="#"
610 FOR X=1 TO 58:READ X1:R1(X)=X1:FOR
 Y=1 TO 4:READ X1:R2(X,Y)=X1:NEXT Y
615 READ X1:M1(X)=X1:READ X1:M2(X)=X1:
READ X1:T1(X)=X1:NEXT X
620 RM=1:A1=1000:A2=1000:W=1:P2=2:POKE
 752,0
800 IF B1=1 THEN GOSUB 20000
```

```
805 IF B1=1 THEN B1=0:GOTO 900
810 POSITION 2,12:? "Want to use an ol
d character ";:INPUT A$:? :IF A$(1,1)<
>"Y" THEN GOSUB 21000:GOTO 900
812 ? "NAME: ";:INPUT NM$
815 ? "STRENGTH: ";:INPUT ST:IF ST>20
OR ST<3 THEN 815
820 ? "DEXTERITY: ";:INPUT DX:IF DX>20
 OR DX<3 THEN 820
825 ? "WOUNDS: ";:INPUT W:W=W/100:IF W
<0.1 OR W>1 THEN 825
830 ? "EXPERIENCE: ";:INPUT EP
832 ? "GOLD: ";:INPUT GP
835 ? "IS (S)HE AN ELF ";:INPUT A$:IF
A$(1,1)="Y" THEN RC=1
836 IF RC=0 THEN ? "IS (S)HE A DWARF "
;:INPUT A$:IF A$(1,1)="Y" THEN RC=2
840 ? "MAGIC ARROWS: ";:INPUT A2:? "NO
RMAL ARROWS: ";:INPUT A1
845 ? "HEALING POTIONS: ";:INPUT PT
846 ? "HOLY WATER: ";:INPUT HW
```

Ask if you want to load in an old game.

```
850 ? "Want to load in an old game ";:
INPUT A$:IF A$(1,1)<>"Y" THEN 900
860 ? "FROM CASSETTE OR DISK ";:INPUT
A$:IF A$(1,1)="C" THEN ? "HIT <RETURN>
 WHEN READY ":GOTO 880
862 IF A$(1,1)<>"D" THEN 860
870 OPEN #2,4,0,"D:QUEST.DAT"
872 FOR X=1 TO 58:INPUT #2;X1:M1(X)=X1
:INPUT #2;X1:M2(X)=X1:INPUT #2;X1:T1(X
)=X1
874 GOTO 884
880 OPEN #2,4,0,"C:QUEST.DAT"
882 FOR X=1 TO 58:INPUT #2;X1:M1(X)=X1
:INPUT #2;X1:M2(X)=X1:INPUT #2;X1:T1(X
)=X1
884 NEXT X:CLOSE #2
```

Marketplace and bargaining routine.

```
900 ? CHR$(125);"GOLD: ";GP
901 ? :? "You're at a market. Prices h
ere are:":?
903 ? "[1] MAGIC ARROW ............ 2
 GOLD  [2] FOUR NORMAL ARROWS ...... 1
 GOLD"
905 ? "[3] HEALING POTION ......... 15
 GOLD  [4] HOLY WATER ............. 3
 GOLD"
910 ? :? "OK, ";NM$;", what do you nee
d ";:INPUT A$:IT=VAL(A$)
911 IF IT>4 OR IT<0 THEN ? CHR$(253);"
I DON'T SELL THAT!":GOTO 910
912 IF IT=0 THEN 990
913 IF IT=1 THEN P1=2
914 IF IT=2 THEN P1=1
915 IF IT=3 THEN P1=15
916 IF IT=4 THEN P1=3
917 ? :? "At ";P1;" GOLD apiece,":? "h
ow many will you buy ";:INPUT A$:NM=VA
L(A$):PRINT
918 IF NM<0 THEN PRINT CHR$(253);"VERY
 FUNNY!!":PRINT "I DON'T BUY THINGS, I
 SELL THEM!!":GOTO 916
920 P1=P1*NM
921 OP=P1
925 ? :? "That comes to ";P1;" GOLD, "
;NM$
930 ? "How much will you give me ";:IN
PUT A:?
935 IF A<OP/10 THEN ? "FORGET IT!!!":G
OTO 901
940 IF A<OP/2 THEN ? "NOT INTERESTED."
:GOTO 901
941 IF A>=P1 THEN ? "YOU GOT A DEAL!":
GOTO 950
942 Y=A/P1:X=RND(0):IF X>Y THEN ? "Not
 interested.":P1=INT((OP+P1)/2):GOTO 9
30
945 P1=INT((P1*2+A)/3):IF P1<=A THEN 9
41
947 ? "How about ";P1;", ";NM$;"?":GOT
O 930
950 IF GP<P1 THEN ? :? "WHAT!! YA CAN'
T PAY YER DEBTS!":? "YOU'LL BE THROWN
INTO PRISON FOR THIS!";CHR$(253):END
955 GP=GP-P1:? "You now have ";GP;" GO
```

```
LD, ";NM$
957 IF IT=4 THEN HW=HW+NM
960 IF IT=1 THEN A2=A2+NM
965 IF IT=2 THEN A1=A1+NM*4
970 IF IT=3 THEN PT=PT+NM
980 GOTO 901
```

Enter dungeon; check for too many arrows.

```
990 ? "OK, ";NM$;", PRESS <RETURN> TO"
:? "ENTER THE DUNGEON!"
991 EL=0
992 IF EP>100 THEN EP=EP-100:EL=EL+100
:FOR X=1 TO 58:M2(X)=M2(X)*1.1:NEXT X:
GOTO 992
993 EP=EP+EL
994 IF EL>500 THEN FOR U=EL TO 500 STE
P -100:FOR X=1 TO 58:M2(X)=M2(X)/1.1:N
EXT X:NEXT U
995 INPUT A$:? CHR$(125):POKE 752,1
996 A3=0:A4=0
997 IF A2>ST*2 THEN A4=A2-ST*2:A2=ST*2
:? "MORE THAN ";ST*2;" MAGIC ARROWS WO
ULD":? "WEIGH YOU DOWN":?
998 IF A1>ST*2 THEN A3=A1-ST*2:A1=ST*2
:? "MORE THAN ";ST*2;" ARROWS WOULD":?
 "WEIGHT YOU DOWN":?
999 FOR X=1 TO 500:NEXT X
```

Upon entering a new room, draw it with its monsters and treasures. If this is room one, give option to leave.

```
1000 POKE 752,1:? CHR$(125):IF B1=0 TH
EN B1=1:GOTO 1005
1001 IF RM<>1 THEN 1005
1002 PRINT "Do you wish to leave the d
ungeon ";:INPUT A$
1003 IF A$(1,1)="Y" THEN 800
1004 ? CHR$(125)
1005 ON R1(RM) GOSUB 10000,11000
1010 IF T1(RM)>0 THEN TX=INT(RND(0)*9)
+16:TY=INT(RND(0)*6)+9:POSITION TX,TY:
PRINT "*";
1015 X5=20:Y5=11
1020 IF I$="W" THEN Y5=22
1022 IF I$="X" THEN Y5=2
1024 IF I$="D" THEN X5=2
1026 IF I$="A" THEN X5=38
1028 POSITION X5,Y5:? YY$;
1030 IF M2(RM)>=1 THEN WX=INT(RND(0)*9
)+16:WY=INT(RND(0)*6)+9
1031 MS=MS(M1(RM))/10
1050 IF M2(RM)>=1 THEN POSITION WX,WY:
PRINT M$(M1(RM),M1(RM));
```

Print player status. Check for wandering monsters.

```
1055 POSITION 2,1:? "ARROWS:";A1;" ";:
POSITION 2,0:? "M ARROWS:";A2;" ";
1060 POSITION 2,21:? "ST=";ST;" DX=";D
X;:POSITION 2,2:? "WOUNDS:";INT(W*100+
0.5);" ";
1061 POSITION 2,3:? "ROOM:";RM;" ";
1062 POSITION 31,17:? "POTIONS:";:POSI
TION 33,18:? PT;" ";
1063 POSITION 31,19:? "H WATER:";:POSI
TION 33,20:? HW;" ";
1065 IF M2(RM)>=1 THEN POSITION 27,0:?
 "MONSTER:";:POSITION 27,1:? MN$((M1(R
M)*11)-10,M1(RM)*11)
1070 IF M2(RM)<1 THEN POSITION 27,0:?
"             ";:POSITION 27,1:? "
        ";
1075 IF M2(RM)>1 THEN POSITION 27,2:?
"NUMBER:";INT(M2(RM));"  ";
1077 IF M2(RM)<=1 THEN POSITION 27,2:?
 "            ";
1080 POSITION 2,17:? "EX:";INT(EP);" "
;:POSITION 37,3:? "W";:POSITION 36,4:?
 "A D";:POSITION 37,5:? "X";
1085 POSITION 2,18:? "GP:";GP;" ";
1086 IF M2(RM)<>0 OR (INT(RND(0)*100)+
1)<>1 THEN 1090
1087 FOR X=1 TO 7:POSITION 4,23:? "WAN
DERING MONSTER!";:FOR Y=1 TO 40:NEXT Y
:POSITION 4,23
1088 ? "                   ";:FOR Y=1 T
O 15:NEXT Y:NEXT X:M2(RM)=INT(RND(0)*3
)+1:M1(RM)=INT(RND(0)*8)+1:GOTO 1030
```

Accept a command from the keyboard and call the appropriate subroutine.

```
1090 A$="":FOR X=1 TO DX*10-EP:IF PEEK
(764)=255 THEN NEXT X:GOTO 1093
1091 GET #3,A:POKE 764,255:A$=CHR$(A)
1093 IF T1(RM)>0 THEN POSITION TX,TY:P
RINT "*";
1095 IF A$="" THEN A$=I$:GOTO 1100
1097 I$=A$
1100 IF I$="W" THEN GOSUB 15100
1105 IF I$="X" THEN GOSUB 15200
1110 IF I$="D" THEN GOSUB 15300
1115 IF I$="A" THEN GOSUB 15400
1120 IF I$="H" THEN I$="":IF PT>0 THEN
 PT=PT-1:W=1
1125 IF I$="M" AND A2>0 THEN I$="":A2=
A2-1:GOSUB 15500
1130 IF I$="N" AND A1>0 THEN I$="":A1=
A1-1:GOSUB 15600
1135 IF I$="F" THEN GOSUB 16000
1140 IF I$="O" THEN GOSUB 17000
1145 IF I$="T" AND HW>0 THEN I$="":HW=
HW-1:GOSUB 18000
```

If there is a monster, move it, and let it attack.

```
1200 IF M2(RM)<1 THEN 1030
1201 IF MS<=0 THEN FOR X=1 TO LEN(K$):
POSITION WX,WY:? K$(X,X);:NEXT X:M2(RM
)=M2(RM)-1:EP=EP+MS(M1(RM)):GOTO 1030
1205 MX=SGN(X5-WX)
1206 IF WX<X5 THEN MX=1
1207 IF WX=X5 THEN MX=0
1210 MY=SGN(Y5-WY)
1215 POSITION WX,WY:? " ";
1220 LOCATE WX+MX,WY,A:POSITION WX+MX,
WY:? CHR$(A);:IF A=32 THEN WX=WX+MX
1225 LOCATE WX,MY+WY,A:POSITION WX,MY+
WY:? CHR$(A);:IF A=32 THEN WY=WY+MY
1230 IF ABS(WX-X5)>1 OR ABS(WY-Y5)>1 T
HEN 1050
1235 X=RND(0):IF X>MS THEN 1050
1240 X=RND(0)*MS
1245 W=W-X:IF W<0 THEN 5000
1250 GOTO 1050
```

End-game routine for the "Great Dungeon In The Sky."

```
5000 FOR X=1 TO LEN(K$):POSITION X5,Y5
:? K$(X,X);:NEXT X:FOR X=1 TO 400:NEXT
 X:? CHR$(125)
5005 ? "WELCOME TO HEAVEN, ";NM$;"!!!"
5010 ? "I HOPE YOU ENJOYED YOUR SHORT
LIFTIME IN WHICH YOU ACCUMULATED ";GP;
" GOLD"
5015 ? "AND ";EP;" EXPERIENCE POINTS."
5020 ? :? :? "WOULD YOU LIKE TO BE REI
NCARNATED AS  A NEW CHARACTER ";:INPUT
 A$:IF A$(1,1)="N" THEN GRAPHICS 0:END
5025 POKE 106,PEEK(106)+5:GRAPHICS 0:R
UN
```

Subroutine to draw a passage/intersection.

```
10000 X1=R2(RM,1)
10010 IF X1>0 THEN FOR X=0 TO 7:POSITI
ON 15,X:PRINT "&";:POSITION 25,X:PRINT
 "&";:NEXT X
10012 IF X1<=0 THEN POSITION 15,7:? "&
&&&&&&&&&&";
10015 X1=R2(RM,2)
10020 IF X1>0 THEN FOR X=15 TO 22:POSI
TION 15,X:PRINT "&";:POSITION 25,X:PRI
NT "&";:NEXT X
10022 IF X1<=0 THEN POSITION 15,15:? "
&&&&&&&&&&&";
10025 X1=R2(RM,3)
10030 IF X1>0 THEN FOR X=25 TO 38:POSI
TION X,7:PRINT "&";:POSITION X,15:PRIN
T "&";:NEXT X
10032 IF X1<=0 THEN FOR X=7 TO 15:POSI
TION 25,X:? "&";:NEXT X
10035 X1=R2(RM,4)
10040 IF X1>0 THEN FOR X=2 TO 15:POSIT
ION X,7:PRINT "&";:POSITION X,15:PRINT
 "&";:NEXT X
10042 IF X1<=0 THEN FOR X=7 TO 15:POSI
```

```
TION 15,X:? "&";:NEXT X
10045 RETURN
```

Subroutine to draw a chamber/room.

```
11000 POSITION 11,4:? "&&&&&";:POSITIO
N 25,4:? "&&&&&";
11010 POSITION 11,18:? "&&&&&";:POSITI
ON 25,18:? "&&&&&";
11012 FOR X=5 TO 7:POSITION 11,X:? "&
                  &";:NEXT X
11014 FOR X=15 TO 17:POSITION 11,X:? "
&                  &";:NEXT X
11015 X1=R2(RM,1)
11020 IF X1>0 THEN FOR X=0 TO 4:POSITI
ON 15,X:PRINT "&";:POSITION 25,X:PRINT
 "&";:NEXT X
11022 IF X1<=0 THEN POSITION 15,4:? "&
&&&&&&&&&&";
11025 X1=R2(RM,2)
11030 IF X1>0 THEN FOR X=18 TO 22:POSI
TION 15,X:PRINT "&";:POSITION 25,X:PRI
NT "&";:NEXT X
11032 IF X1<=0 THEN POSITION 15,18:? "
&&&&&&&&&&&";
11035 X1=R2(RM,3)
11040 IF X1>0 THEN POSITION 29,7:PRINT
 "&&&&&&&&&&";:POSITION 29,15:PRINT "&
&&&&&&&&&";
11042 IF X1<=0 THEN FOR X=7 TO 15:POSI
TION 29,X:? "&";:NEXT X
11045 X1=R2(RM,4)
11050 IF X1>0 THEN POSITION 2,7:PRINT
"&&&&&&&&&&";:POSITION 2,15:PRINT "&&&
&&&&&&&";
11052 IF X1<=0 THEN FOR X=7 TO 15:POSI
TION 11,X:? "&";:NEXT X
11055 RETURN
```

Subroutine to move player about on the screen.

```
15100 IF Y5=1 THEN 15105
15102 LOCATE X5,Y5-1,M:POSITION X5,Y5-
1:? CHR$(M):IF M<>32 THEN RETURN
15105 POSITION X5,Y5:? " ";
15110 Y5=Y5-1:IF Y5<1 THEN RM=R2(RM,1)
:GOTO 1000
15120 POSITION X5,Y5:? YY$;:RETURN
15200 IF Y5=21 THEN 15205
15202 LOCATE X5,Y5+1,M:POSITION X5,Y5+
1:? CHR$(M):IF M<>32 THEN RETURN
15205 POSITION X5,Y5:? " ";
15210 Y5=Y5+1:IF Y5>21 THEN RM=R2(RM,2
):GOTO 1000
15220 GOTO 15120
15300 IF X5>37 THEN 15305
15302 LOCATE X5+1,Y5,M:POSITION X5+1,Y
5:? CHR$(M):IF M<>32 THEN RETURN
15303 LOCATE X5+2,Y5,M:POSITION X5+2,Y
5:? CHR$(M):IF M<>32 THEN RETURN
15305 POSITION X5,Y5:? " ";
15310 X5=X5+2:IF X5>37 THEN RM=R2(RM,3
):GOTO 1000
15320 GOTO 15120
15400 IF X5<3 THEN 15405
15402 LOCATE X5-1,Y5,M:POSITION X5-1,Y
5:? CHR$(M):IF M<>32 THEN RETURN
15403 LOCATE X5-2,Y5,M:POSITION X5-2,Y
5:? CHR$(M):IF M<>32 THEN RETURN
15405 POSITION X5,Y5:? " ";
15410 X5=X5-2:IF X5<3 THEN RM=R2(RM,4)
:GOTO 1000
15420 GOTO 15120
```

Normal-arrow firing routine.

```
15500 GOSUB 15699
15505 X=RND(0)/2:IF RC=1 THEN X=X-0.1
15506 IF RC=2 THEN X=X+0.1
15507 X=X-(EP/1000)
15510 X=X-0.2
15511 X=X-(DX/100)
15515 IF X>W THEN RETURN
15520 X=RND(0):IF RC=1 THEN X=X+0.2
15522 IF RC<>1 THEN X=X+0.1
15523 IF RC=2 THEN X=X+0.1
15525 MS=MS-X:RETURN
15599 RETURN
```

Magic-arrow firing routine.

```
15600 GOSUB 15699
15601 IF M1(RM)=8 THEN RETURN
15605 X=RND(0)/2:IF RC=1 THEN X=X-0.1
15606 X=X-(DX/100)
15607 IF RC=2 THEN X=X+0.1
15608 X=X-(EP/1000)
15610 IF X>W THEN RETURN
15620 X=RND(0):IF RC=1 THEN X=X+0.1
15621 IF RC=2 THEN X=X-0.1
15625 MS=MS-X:RETURN
15698 RETURN
```

Calculate monster range, aim, and shoot arrow.

```
15699 IF WX=0 THEN WX=31:IF WY=0 THEN
WY=8
15700 X6=X5:Y6=Y5-1:X7=WX:Y7=WY
15701 IF X6=X7 THEN SL=0:X8=X7:X9=X6
15702 IF X6<>X7 THEN SL=(Y6-Y7)/(X6-X7
):X8=X6:X9=X7
15703 GOTO 15708
15705 IF X6>X7 THEN SL=(Y6-Y7)/(X6-X7)
:X8=X6:X9=X7
15706 IF X7>X6 THEN SL=(Y7-Y6)/(X7-X6)
:X8=X7:X9=X6
15707 IF X7=X6 THEN SL=0:X8=X7:X9=X6
15708 Y8=Y6:Y9=Y7
15709 Y=Y8
15710 SL=SL*SGN(Y8-Y9):IF Y6<Y7 THEN S
L=-SL
15711 IF X6>X7 THEN SL=-SL
15712 FOR X=X8 TO X9 STEP SGN(X9-X8+0.
01):FOR XX=1 TO 5:NEXT XX
15713 IF Y>22 OR Y<1 OR X>37 OR X<3 TH
EN NEXT X:GOTO 15750
15715 LOCATE X,Y,A:IF A=38 THEN X9=X-1
:GOTO 15750
15720 POSITION X,Y:PRINT "+";:Y=Y+SL:N
EXT X
15750 Y=Y8:FOR X=X8 TO X9 STEP SGN(X9-
X8+0.01):POSITION X,Y:PRINT " ";:Y=Y+S
L:NEXT X
15760 RETURN
```

Subroutine for close combat with a monster.

```
16000 IF ABS(X5-WX)>1 OR ABS(Y5-WY)>1
THEN RETURN
16001 IF M1(RM)=8 THEN RETURN
16002 IF M1(RM)=7 THEN RETURN
16003 IF M1(RM)=6 THEN W=W-0.05
16005 X=RND(0):IF RC=0 THEN X=X-0.1
16006 X=X-(DX/100)
16007 IF RC=2 THEN X=X-0.3
16008 X=X-(EP/1000)
16010 IF X>W THEN RETURN
16015 X=RND(0):IF RC=0 THEN X=X+0.1
16016 X=X+(ST/100)
16017 IF RC=2 THEN X=X+0.2
16020 MS=MS-X:RETURN
```

Subroutine for opening a treasure chest.

```
17000 IF ABS(TX-X5)>1 THEN RETURN
17005 IF ABS(TY-Y5)>1 THEN RETURN
17010 POSITION TX,TY:? " ";
17011 TX=0:TY=0
17015 POSITION 12,23:? T$((T1(RM)*23)-
22,T1(RM)*23);:FOR X=1 TO 120:NEXT X
17020 FOR X=12 TO 34:POSITION X,23:? "
 ";:NEXT X
17021 IF T1(RM)=10 THEN PT=PT+1:GOTO 1
7026
17022 IF T1(RM)=11 THEN A2=A2+10:GOTO
17026
17023 IF T1(RM)=12 THEN A1=A1+10:GOTO
17026
17024 TS(T1(RM))=TS(T1(RM))+1
17025 GP=GP+GP(T1(RM))
17026 EP=EP+EP(T1(RM)):T1(RM)=0
17030 RETURN
```

Subroutine to throw a flask of holy water.

```
18000 M=M1(RM):IF M=2 OR M=5 OR M=7 TH
EN RETURN
18005 GOSUB 15699:POSITION WX,WY:? " "
;:WA=WX:WB=WY:WX=X5:WY=Y5:GOSUB 16000:
WX=WA:WY=WB:POSITION X5,Y5:? YY$;:RETU
RN
```

End-of-game procedures, such as saving games and printing out information on the player's character.

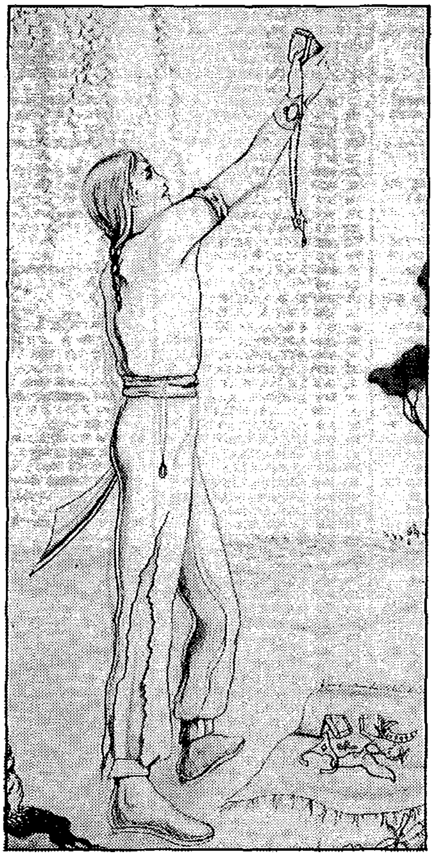
```
20000 PRINT "WOULD YOU LIKE TO SEE THE
 TREASURES  YOU RETRIEVED FROM THE DU
NGEON";:INPUT A$
20005 IF A$(1,1)="Y" THEN FOR X=1 TO 9
:? STR$(X);". ";T$((X*23)-22,X*23);CHR
$(127);TS(X):NEXT X
20010 FOR X=1 TO 9:TS(X)=0:NEXT X
20011 A1=A1+A3:A2=A2+A4
20015 ? :? "WOULD YOU LIKE TO SAVE THI
S GAME ";:INPUT A$
20017 IF A$(1,1)<>"Y" THEN 20028
20018 ? "CASSETTE OR DISK ";:INPUT A$:
IF A$(1,1)="C" THEN ? "HIT <RETURN> WH
EN READY ":GOTO 20024
20019 IF A$(1,1)<>"D" THEN 20018
20020 OPEN #2,8,0,"D:QUEST.DAT"
20021 FOR X=1 TO 58:PRINT #2;M1(X):PRI
NT #2;M2(X):PRINT #2;T1(X)
20022 GOTO 20027
20024 OPEN #2,8,0,"C:QUEST.DAT"
20025 FOR X=1 TO 58:PRINT #2;M1(X):PRI
NT #2;M2(X):PRINT #2;T1(X)
20027 NEXT X:CLOSE #2:? "SAVE COMPLETE
."
20028 ? "Would yOu like to stop now ";
:INPUT A$:IF A$(1,1)<>"Y" THEN RETURN
20030 ? "OK. So that you can use this
characteragain at a later time:"
20035 ? "NAME: ";NM$;" RACE: ";:IF RC=
0 THEN ? "HUMAN"
20036 IF RC=1 THEN ? "ELF"
20037 IF RC=2 THEN ? "DWARF"
20045 ? "HEALING POTIONS: ";PT
20046 ? "HOLY WATER: ";HW
20050 ? "ARROWS: ";A1;"  MAGIC ARROWS
: ";A2
20055 ? "GOLD: ";GP;"  EXPERIENCE: ";
EP
20060 ? "STRENGTH: ";ST;"  DEXTERITY:
 ";DX
20065 ? :? "Would you like to try agai
n":? "as a *NEW* character ";:INPUT A$
:IF A$(1,1)="Y" THEN 5025
20099 GRAPHICS 0:? "COME QUESTING AGAI
N SOMETIME!!":END
```

Subroutine to create new characters.

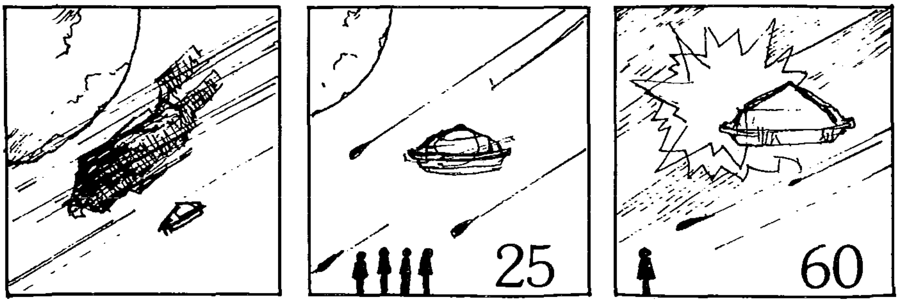
```
21000 ? "OK, I'll make you one.":FOR X
=1 TO 100:NEXT X:GP=INT(RND(0)*20)+6:S
T=INT(RND(0)*17)+4:DX=INT(RND(0)*17)+4
21005 RC=INT(RND(0)*3):A1=3:A2=INT(RND
(0)*10)+1:PT=INT(RND(0)*3)+2:HW=INT(RN
D(0)*5)+1:EP=0:W=1
21010 ? "STRENGTH: ";ST;"  DEXTERITY:
 ";DX
21015 ? "GOLD: ";GP;"  HEALING POTION
S: ";PT
21020 ? "HOLY WATER: ";HW;"  RACE: ";
:IF RC=0 THEN ? "HUMAN"
21021 IF RC=1 THEN ? "ELF"
21022 IF RC=2 THEN ? "DWARF"
21025 ? "ARROWS: ";A1;"  MAGIC ARROWS
: ";A2
21030 ? :? "What will you name this ch
aracter":INPUT NM$:? :? "HAVE A FUN QU
EST, ";NM$;"!!!!"
21040 FOR X=1 TO 200:NEXT X:? CHR$(125
):RETURN
```

Redefine "&" as a red block, and "@" as a man character.

```
30000 POKE 106,PEEK(106)-5:GRAPHICS 0:
? "INITIALIZING . . .":SETCOLOR 2,7,0
30010 START=(PEEK(106)+1)*256
30020 DIM XFR$(38):XFR$=".............
........................":Z=USR(ADR(X
FR$)):RESTORE 30050
30021 REM TO TYPE XFR$ IN LINE 30020,
REPLACE THE STRING OF PERIODS WITH THE
STRING DESCRIBED IN THE DOCUMENTATION
30030 POKE 756,START/256
30040 FOR X=0 TO 7:POKE X+START+48,85:
NEXT X:FOR X=0 TO 7:READ X1:POKE X+STA
RT+256,X1:NEXT X:RETURN
30050 DATA 152,216,255,27,25,60,102,23
1
```



```
1 GOSUB 30000:? CHR$(125):POKE 752,1:P
OSITION 15,6:? " QUEST 2 "
2 ? :? "QUEST 2 was written by Brian R
eynolds":OPEN #3,4,0,"K"
4 REM QUEST 2 DATA BASE BY T. WENRICH
115 DATA 2,41,2,36,11,0,0,1
120 DATA 2,1,8,3,12,1,1,2
125 DATA 1,0,0,4,2,3,2,7
130 DATA 2,0,5,0,3,2,1,2
135 DATA 1,4,6,9,0,6,3,10
140 DATA 2,5,0,0,7,4,1,5
145 DATA 1,0,0,6,8,7,1,6
150 DATA 1,2,0,7,0,5,2,3
155 DATA 1,0,0,10,5,1,3,7
160 DATA 2,0,0,53,9,7,3,6
165 DATA 1,0,12,1,35,2,1,2
170 DATA 2,11,13,2,0,4,2,1
175 DATA 1,12,14,0,0,5,2,4
180 DATA 2,13,0,15,0,7,4,8
185 DATA 1,0,0,16,14,3,1,1
190 DATA 2,0,23,17,15,6,2,7
195 DATA 2,0,19,18,16,8,1,1
200 DATA 1,0,20,0,17,1,2,4
205 DATA 2,17,0,20,0,6,3,8
210 DATA 2,18,21,0,19,4,1,1
215 DATA 2,20,0,0,22,5,2,12
220 DATA 1,0,0,21,23,1,2,3
225 DATA 2,16,0,22,24,7,1,1
240 DATA 1,0,0,23,25,8,1,2
250 DATA 2,0,0,24,26,8,4,9
260 DATA 1,27,0,25,0,2,1,1
270 DATA 2,30,26,0,28,5,2,7
280 DATA 2,29,0,27,45,6,1,2
290 DATA 2,0,28,30,0,7,5,9
300 DATA 1,31,27,0,29,4,2,5
310 DATA 1,32,30,0,0,1,2,3
320 DATA 2,0,31,34,33,5,3,8
330 DATA 1,0,0,32,0,3,1,4
340 DATA 2,35,0,0,32,1,1,1
350 DATA 2,36,34,11,0,3,2,6
360 DATA 2,40,35,37,1,8,5,1
370 DATA 1,39,0,0,36,2,1,2
380 DATA 2,0,0,0,39,6,3,8
390 DATA 1,46,37,38,0,1,2,5
400 DATA 1,0,36,0,41,7,1,3
410 DATA 1,42,1,40,0,1,1,1
420 DATA 2,43,41,0,0,4,1,2
430 DATA 1,50,42,0,0,5,2,5
440 DATA 2,48,0,46,0,2,6,8
450 DATA 2,0,0,28,47,6,3,8
460 DATA 2,47,39,0,44,3,2,7
470 DATA 1,0,46,45,48,1,2,1
480 DATA 2,49,44,47,0,4,1,2
490 DATA 1,0,48,0,50,1,1,1
500 DATA 2,0,43,49,51,8,3,8
510 DATA 1,0,52,50,0,5,1,3
520 DATA 2,51,54,0,0,2,2,4
530 DATA 2,0,0,54,10,7,4,9
540 DATA 1,52,55,0,53,4,5,7
550 DATA 1,54,0,0,56,7,1,11
560 DATA 2,0,0,55,57,1,2,6
570 DATA 1,0,58,56,0,5,3,7
580 DATA 2,57,0,0,0,8,10,9
```



SPACE RESCUE

by Matt Rutter

***Space Rescue* is an arcade-style game for a 16K Atari with one joystick.**

The year is 2086. Just a few months ago, the United States launched an exploration party to search the planet Arcturus III for any signs of life. Our radar, however, has just picked up a huge meteor storm headed straight for that solar system which threatens the lives of all the people there. You are their only hope of survival. You must launch your two-person rocket from the mother ship orbiting around the planet, land at the one landing pad, rescue the people stranded there one at a time, and then return to the mother ship — all while trying to avoid crashing your fragile rocket into one of the deadly meteors which can easily destroy it.

When the game starts, the computer will show the mother ship moving back and forth at the top of the screen with a cluster of meteors right below it. When you think that the ship is right over a path through the meteors, press the fire button on the joystick to launch your rocket. Then you must guide the rocket down to the landing pad by moving your joystick, being careful not to collide with any of the meteors. To slow down, press the fire button to apply thrust. After you have landed, one of the people will run over to your ship, and then you must make your ascent to the mother ship and dock with her, again carefully avoiding the deadly meteors. On the way up, pressing the fire button will launch a missile which can destroy a meteor, for which you can receive points.

If one of your rockets collides with a meteor, then it is destroyed. The game continues until all three of your rockets are destroyed, at which point the game is over. When all of the people are gone from the bottom of the screen, the computer will award you 50 bonus points for each person safely brought to the mother ship, and will then give you six more people to save. If you succeed in rescuing all six, you will be awarded one bonus ship. During the game, the score is displayed at the bottom of the screen underneath the landing pad.

Variables

A, B: Used in determining whether or not a collision has occurred.
D: Direction in which mother ship is travelling (1 or -1).
D1: Difficulty level.
G: Missile-launched flag.
L: Position of landing pad at bottom of screen.
P: Value returned from input commands.

P1: Number of people left at bottom of screen.
P2: Number of people safely brought to mother ship.
S: Position of mother ship at top of screen.
SC: Score.
SL: Number of ships left.
U: Rocket-going-up flag.
X, Y: Position of rocket.
XM, YM: Position of missile.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS        'Space Rescue'       SS
SS      Author: Matt Rutter    SS
SS      Copyright (c) 1982     SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Lines 30, 50, 60, 80, 280, and 370 contain graphics and/or control characters.

```
10 GOSUB 1100:GOSUB 1000
```

Initialize variables. Set character base register (POKE location 756) to beginning of altered character set. Display six people at the bottom of the screen.

```
15 D1=0:SC=0:SL=3
20 GRAPHICS 0:POKE 756,R:POKE 752,1:SE
TCOLOR 2,0,0:S=34:D=-1:P1=6:P2=0
25 FOR I=21 TO 23:POSITION 0,I:? "f";:
POSITION 38,I:? "f";:NEXT I
```

Draw the landing pad, and put the meteors on the screen.

```
30 L=INT(23*RND(1)+5):POSITION L,21:?
"__":POSITION L-1,22:? "____":POSITI
ON L-2,23:? "______";:B=0:G=0
40 FOR I=1 TO D1*10+30:POSITION RND(1)
*35+1,RND(1)*15+3:? CHR$(20);:NEXT I:P
OSITION L,23:? SC;:POKE 77,0
45 FOR AZ=1 TO SL:SOUND 0,37,10,10:FOR
 W=1 TO 100:NEXT W:SOUND 0,0,0,0:NEXT
AZ
```

Move mother ship back and forth.

```
50 POSITION S,0:? "     =+++++     =++
++   ";:S=S+D
51 REM LINE 50: NON-SPACE CHARACTERS
ARE CONTROL CHARACTERS
60 POSITION S,0:? "gnnnf=+++++b";CHR$(
97);CHR$(98);CHR$(99);"v=++++mmm";:IF
S<3 OR S>33 THEN D=-D
61 REM LINE 60: CHARACTERS ARE CONTROL
CHARACTERS
65 IF STRIG(0)=0 THEN 80
70 FOR W=1 TO 25:NEXT W:GOTO 50
80 POSITION S+2,2:? " ":FOR W=1 TO 50:
NEXT W:POSITION S,2:? "m   m":X=S+2:Y=
1
81 REM LINE 80: NON-SPACE CHARACTERS
ARE CONTROL CHARACTERS
```

Check for player input.

```
100 U=0:X1=X:P=STICK(0):IF P<12 AND Y>
1.5 THEN GOSUB 300
120 Y1=Y+1:P=STRIG(0):IF P=0 THEN Y1=Y
+0.5
```

Check for collision.

```
130 FOR I=X1-1 TO X1+1:LOCATE I,Y1,A:I
F A=20 OR A=160 THEN 500
133 IF B=20 OR B=160 THEN 500
135 NEXT I
```

Move ship, and display flame if appropriate.

```
140 POSITION X-1,Y:? "   ":POSITION X,
Y+1:? " ":X=X1:Y=Y1:POSITION X-1,Y:? "
abc"
145 LOCATE X,Y+1,B:IF P=0 THEN POSITIO
N X,Y+1:? "d":SOUND 1,20,8,15
147 IF P=1 THEN SOUND 1,0,0,0
```

Check for successful landing.

```
150 IF X=L+1 AND INT(Y+0.5)=20 THEN PO
SITION X,Y+1:? CHR$(160):SC=SC+50:POSI
TION L,23:? SC;:GOTO 400
```

```
155 IF Y>20 THEN 500
160 GOTO 100
```

Erase old meteors, and display new ones.

```
200 X=L+1:Y=20:COLOR 32:FOR I=3 TO 18:
PLOT 1,I:DRAWTO 37,I:NEXT I
205 FOR I=1 TO D1*15+40:POSITION RND(1
)*35+1,RND(1)*15+3:? CHR$(20);:NEXT I:
C=0
207 POSITION L,21:? "   "
```

Make launching sound, check for input, and move ship up one space.

```
210 SOUND 0,5*Y+90,8,8:X1=X:Y1=Y:P=STI
CK(0):IF P<12 THEN GOSUB 300
220 C=C+1:IF C=3 THEN C=0:Y1=Y-1
```

Check for collision.

```
225 FOR I=X1-1 TO X1+1:LOCATE I,Y1,A:I
F A=20 THEN 500
227 NEXT I
```

Move rocket, and check for input.

```
230 POSITION X-1,Y:? "   ":POSITION X,
Y+1:? " ";:X=X1:Y=Y1:POSITION X-1,Y:?
"abc":POSITION X,Y+1:? "d";
240 IF STRIG(0)=0 AND G=0 AND Y>4 THEN
 GOSUB 340
250 IF G=1 THEN GOSUB 350
255 IF Y=1 THEN 270
260 GOTO 210
```

Check for successful docking with mother ship.

```
270 IF X<>S+2 THEN 500
275 P2=P2+1
280 SOUND 0,0,0,0:POSITION S,2:? " aaa
 ":COLOR 32:FOR I=3 TO 23:PLOT 1,I:DRA
WTO 37,I:NEXT I
281 REM LINE 280: NON-SPACE CHARACTERS
ARE CONTROL CHARACTERS
```

If all the people are gone from the bottom of the screen, increase the difficulty, and branch to the bonus routine.

```
285 IF P1=0 THEN P1=6:D1=D1+1:GOTO 650
290 GOTO 30
```

Subroutine to move ship according to input.

```
300 IF P>8 AND P<12 AND X>2 THEN X1=X-
1
310 IF P<8 AND X<36 THEN X1=X+1
320 RETURN
```

Launch missile.

```
340 G=1:XM=X:YM=Y:FOR I=10 TO 100 STEP
 10:SOUND 1,I,10,10:NEXT I:SOUND 1,0,0
,0
```

Move missile and check for collision with meteors.

```
350 POSITION XM,YM:? " ";:YM=YM-1
360 LOCATE XM,YM,A:IF A=20 THEN POSITI
ON XM,YM:? " ";:G=0:SC=SC+20:POSITION
L,23:? SC;:RETURN
370 POSITION XM,YM:? "|":IF YM<4 THEN
POSITION XM,YM:? " ":G=0
380 RETURN
```

Successful-landing routine. Wave arms of the next person, and move him over to the rocket.

```
400 SOUND 1,0,0,0:Y=27-P1:X=38:X1=34:X
2=L+5:IF P1<4 THEN Y=24-P1:X=0:X1=1:X2
=L-3
410 FOR I=1 TO 8:FOR J=101 TO 103:POSI
TION X,Y:? CHR$(J);:FOR W=1 TO 30:NEXT
 W:NEXT J
420 POSITION X1,Y:? "YAY!";:IF I/2=INT
(I/2) THEN POSITION X1,Y:? "    ";
430 NEXT I:POSITION X,Y:? " ";
440 FOR I=X+SGN(X1-X) TO X2 STEP SGN(X
2-X)
```

```
450 FOR J=101 TO 103 STEP 2:POSITION I
,23:? CHR$(J);:FOR W=1 TO 20:NEXT W
455 SOUND 0,90,8,15:SOUND 0,0,0,0:NEXT
 J:POSITION I,23:? " ";:NEXT I
460 X=I+SGN(X1-X2):Y=23:FOR I=1 TO 2:X
=X+SGN(L-X1):Y=Y-1
465 FOR J=101 TO 103 STEP 2:POSITION X
,Y:? CHR$(J);:FOR W=1 TO 20:NEXT W:NEX
T J:POSITION X,Y:? " ";:NEXT I
470 P1=P1-1:U=1:GOTO 200
```

Explosion routine.

```
500 SOUND 0,0,0,0:SOUND 1,75,8,15
520 X1=X-2:X2=X+1:X0=X:XS=2:V0=15:V1=1
5:V2=15:POSITION X-1,Y:? "   ":POSITIO
N X,Y+1:? " "
530 FOR I=Y TO 22:POSITION X1,I:? "ah"
;:POSITION X2,I:? "ic";:IF U=1 THEN PO
SITION X0,I:? "e";
540 SOUND 0,20,8,V0:SOUND 1,40,8,V1:SO
UND 2,70,8,V2:V0=V0*0.7:V1=V1*0.75:V2=
V2*0.78:XS=XS-0.2:IF XS<0 THEN XS=0
550 FOR W=1 TO 30:NEXT W:POSITION X1,I
:? "  ";:POSITION X2,I:? "  ";:IF U=1
THEN POSITION X0,I:? " ";
560 X1=X1-XS:IF X1<1 THEN X1=36
570 X2=X2+XS:IF X2>36 THEN X2=1
580 NEXT I:FOR I=0 TO 3:SOUND I,0,0,0:
NEXT I:SL=SL-1:IF SL=0 THEN 600
590 GOTO 280
```

End of game. Print score and game-over message, and wait for input.

```
600 GRAPHICS 0:POKE 752,1:POSITION 16,
10:? "GAME OVER":POSITION 12,12:? "YOU
R SCORE IS ";SC
610 POSITION 8,14:? "PUSH START TO PLA
Y AGAIN"
620 IF PEEK(53279)=6 THEN 15
630 GOTO 620
```

Bonus routine. Award 50 points for each person safely brought to the mother ship.

```
650 FOR I=3 TO 23:PLOT 1,I:DRAWTO 37,I
:NEXT I
660 POSITION 16,10:? "*BONUS*":POSITIO
N 15,12:? "SCORE=";SC
670 FOR I=1 TO P2:POSITION 12+I*2,14:?
 "f":SC=SC+50:POSITION 21,12:? SC
680 SOUND 0,I*20+40,10,10:FOR W=1 TO 5
0:NEXT W:NEXT I
690 FOR I=P2 TO 1 STEP -1:SOUND 0,I*20
+40,10,10:FOR W=1 TO 50:NEXT W:NEXT I:
SOUND 0,0,0,0:IF P2<6 THEN 699
```

Bonus-ship routine. Award a bonus ship if all six people are rescued.

```
693 POSITION 10,16:? "*** BONUS SHIP!
***";:SL=SL+1:FOR P2=1 TO 4:FOR AZ=80
TO 185 STEP 6
696 SOUND 0,AZ,10,10:SOUND 0,AZ+50,10,
10:NEXT AZ:NEXT P2
699 FOR W=11 TO 111 STEP 5:FOR AZ=W TO
 W-7 STEP -1:SOUND 0,AZ,10,10:NEXT AZ:
NEXT W:SOUND 0,0,0,0:GOTO 20
```

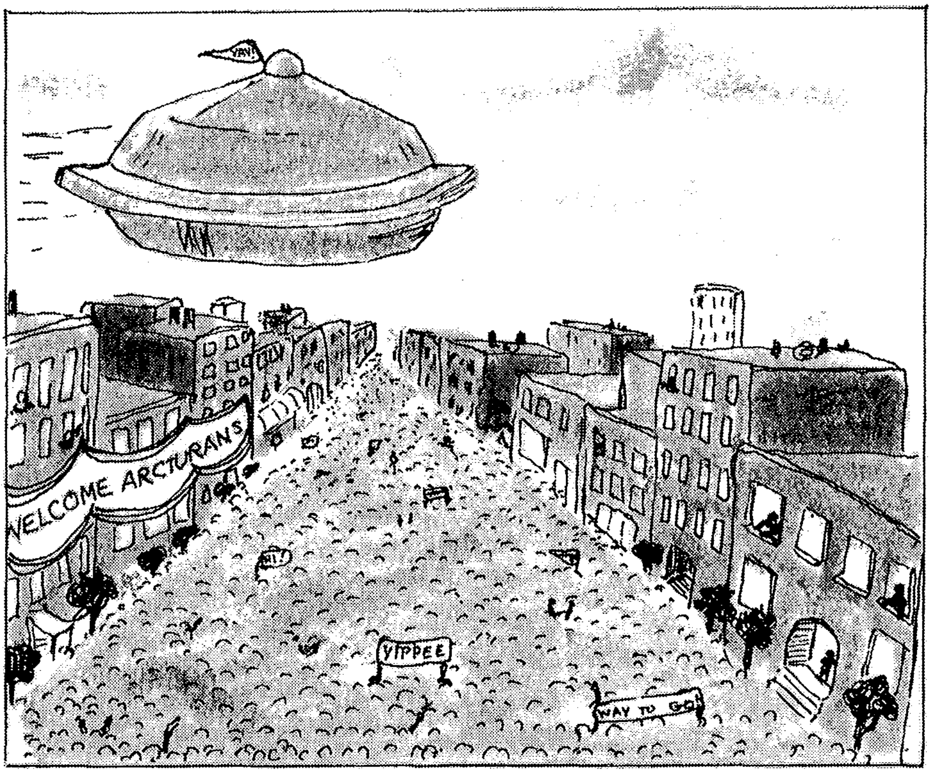
Alter the character set. First transfer the old character set (beginning at ROM address 57344) to RAM, beginning 8 pages (2048 bytes) from the top of memory. Then poke the new characters from DATA statements into the new character set in RAM. This alters the lower-case characters "a" to "i."

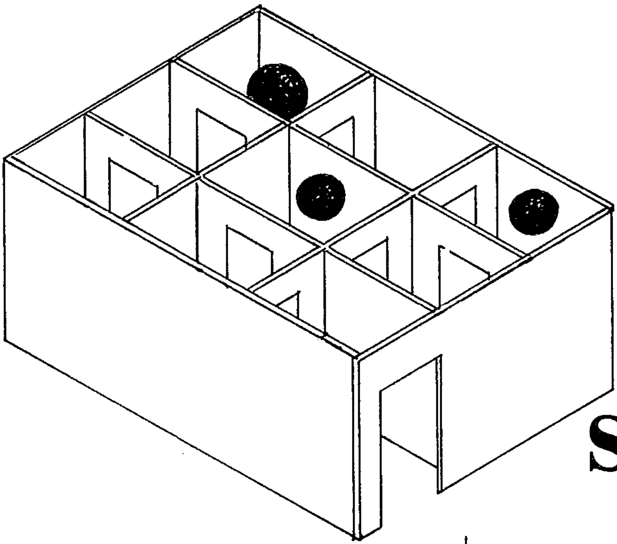
```
1000 R=PEEK(106)-8:RM=R*256
1010 C=0:FOR I=0 TO 1023 STEP 30:C=C+1
:IF C=3 THEN C=0
1020 SETCOLOR C,1,8:FOR J=I TO I+30:PO
KE RM+J,PEEK(57344+J):NEXT J:SETCOLOR
C,0,0:NEXT I:SETCOLOR 0,1,8
1030 FOR I=0 TO 8:FOR J=0 TO 7:READ A:
POKE RM+(97+I)*8+J,A:NEXT J:NEXT I:RET
URN
1040 REM DATA FOR NEW CHARACTERS, IN
GROUPS OF 8
1050 DATA 0,0,12,12,12,15,15,12,24,24,
24,60,255,255,255,0,0,0,48,48,48,240,2
40,48
```

```
1060 DATA 255,255,126,126,60,60,24,24,
219,126,60,24,24,60,102,195,24,24,255,
24,24,60,102,195
1070 DATA 24,24,60,126,219,60,102,195,
48,48,48,112,240,240,240,0,12,12,12,14
,15,15,15,0
1100 GRAPHICS 3+16:FOR I=0 TO 2:SETCOL
OR I,0,0:NEXT I
1105 C=0:FOR I=0 TO 39:C=C+1:IF C=4 TH
EN C=1
```

Draw the opening display on the screen.

```
1110 COLOR C:PLOT I,0:PLOT 39-I,21:NEX
T I
1120 C=0:FOR I=0 TO 21:C=C+1:IF C=4 TH
EN C=1
1130 COLOR C:PLOT 39,I:PLOT 0,21-I:NEX
T I
1140 COLOR 1:PLOT 9,4:DRAWTO 5,4:DRAWT
O 5,6:DRAWTO 9,6:DRAWTO 9,9:DRAWTO 5,9
:PLOT 11,9:DRAWTO 11,4:DRAWTO 15,4
1142 DRAWTO 15,7:DRAWTO 11,7:PLOT 17,9
:DRAWTO 17,4:DRAWTO 21,4:DRAWTO 21,9:P
LOT 17,6:DRAWTO 21,6:PLOT 27,4
1144 DRAWTO 23,4:DRAWTO 23,9:DRAWTO 27
,9:PLOT 33,4:DRAWTO 29,4:DRAWTO 29,9:D
RAWTO 33,9:PLOT 29,6:DRAWTO 33,6
1150 PLOT 2,17:DRAWTO 2,12:DRAWTO 6,12
:DRAWTO 6,15:DRAWTO 2,15:PLOT 5,16:PLO
T 6,17:PLOT 12,12:DRAWTO 8,12
1152 DRAWTO 8,17:DRAWTO 12,17:PLOT 8,1
4:DRAWTO 12,14:PLOT 18,12:DRAWTO 14,12
:DRAWTO 14,14:DRAWTO 18,14
1154 DRAWTO 18,17:DRAWTO 14,17:PLOT 24
,12:DRAWTO 20,12:DRAWTO 20,17:DRAWTO 2
4,17:PLOT 26,12:DRAWTO 26,17
1156 DRAWTO 30,17:DRAWTO 30,12:PLOT 36
,12:DRAWTO 32,12:DRAWTO 32,17:DRAWTO 3
6,17:PLOT 32,14:DRAWTO 36,14
1160 RETURN
```





MAZE SEARCH

by Dave Bohlke

Maze Search is an arcade-style game for an Atari with a joystick in slot 1 and 8K RAM.

Has your mind been uncontrollably wandering lately? Then try putting it back in line with a game that demands intense concentration. In *Maze Search,* you not only need to control your moving cursor to intercept the 24 targets, but you must also diligently plan ahead to be able to negotiate the maze. Only the very best players will capture all of the targets.

Each game begins with your Atari generating a random maze. Every section within the maze has only one entrance/exit to the outside corridor — which is your access to the other sections of the maze. After the maze is completed, 24 blue targets will be placed at random in the maze. The object of *Maze Search* is to move your green cursor and run over the blue targets. Several players can compete in turn to determine who can intercept the most targets.

There is a time limit to each game, which limit you can adjust by changing the value of variable CT in line 600. The time remaining in the game is indicated by a green bar at the right side of the screen. To use your time most efficiently, keep your cursor moving!

Each time you intercept a target, it will be displayed along the left side of the screen. When the game is over, press the fire button on your joystick for another game.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS        'Maze  Search'       SS
SS     Author: Dave Bohlke     SS
SS      Copyright (c) 1982     SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS

1 GOTO 100
```

Direction changes for the STICK(0) commands. See p. 60 of the Reference Manual.

```
5 GOTO 7
6 GOTO 14
7 X=X+1:RETURN
9 GOTO 13
10 GOTO 11
11 X=X-1:RETURN
13 Y=Y+1:RETURN
14 Y=Y-1:RETURN
```

Full-screen, mode 3 graphics.

```
100 GRAPHICS 3+16
```

Outline of maze.

```
120 COLOR 1:PLOT 4,0:DRAWTO 36,0:DRAWT
O 36,22:DRAWTO 4,22:DRAWTO 4,0
```

Arrays M and N hold the locations of all the intersections in the maze.

```
200 DIM M(200),N(200)
```

K is the intersection counter. M and N are the screen location of the intersection.

```
210 K=0:M=18:N=10
```

Plot the initial intersection.

```
220 PLOT M,N
```

Lines 240-390: Loop for maze construction.

Test whether next intesection is filled in.

```
240 LOCATE M+2,N,X:IF X=0 THEN 250
242 LOCATE M-2,N,X:IF X=0 THEN 250
244 LOCATE M,N+2,X:IF X=0 THEN 250
246 LOCATE M,N-2,X:IF X=0 THEN 250
```

Next intersection is used, so jump to line 370 to get the previous intersection location, and decrement the counter. K = 0 means we're done.

```
248 GOTO 370
```

When there is at least one open adjacent intersection, pick a random direction, D.

```
250 D=INT(RND(0)*4)+251:GOTO D
```

Offsets for particular directions (M1 and N1).

```
251 M1=-1:N1=0:GOTO 300
252 M1=0:N1=1:GOTO 300
253 M1=1:N1=0:GOTO 300
254 M1=0:N1=-1
```

See whether the intersection at the random location is open; jump to line 250 if not.

```
300 LOCATE M+M1*2,N+N1*2,X:IF X=1 THEN
 250
```

Plot to the open intersection.

```
310 PLOT M+M1,N+N1:PLOT M+M1*2,N+N1*2
```

Sound and random color.

```
320 SOUND 0,M+N,10,4:SETCOLOR 0,RND(0)
*15,10:SETCOLOR 4,RND(0)*15,8
```

Adjust M and N to the new intersection. Increment K and M(K) and N(K) as the the intersection location. Branch to start the process again.

```
330 M=M+M1*2:N=N+N1*2:K=K+1:M(K)=M:N(K
)=N:GOTO 240
```

All directions are blocked, so back up to the previous intersection.

```
370 M=M(K):N=N(K):K=K-1
```

Check for maze completion.

```
380 IF K=0 THEN 400
```

Not finished, so branch to check previous intersection.

```
390 GOTO 240
```

Clear dimensions; they are no longer needed.

```
400 CLR
```

Player starting location.

```
405 M=11:N=11
```

Set colors.

```
410 SETCOLOR 0,1,10:SETCOLOR 4,4,6
420 SETCOLOR 2,8,8:COLOR 2:PLOT M,N
```

Plot 24 target blocks.

```
430 SETCOLOR 1,12,8:FOR I=1 TO 24:GOSU
B 900:NEXT I
```

Lines 500-700: Main game loop.
Check stick; if not pressed, branch to line 600.

```
500 S=STICK(0):IF S=15 THEN 600
```

Save position (M, N) in scratch variables X and Y. GOSUB S will be 5 to 14 for direction offset.

```
510 X=M:Y=N:GOSUB S
```

Blank player's block. Locate new position (X, Y).

```
520 COLOR 0:PLOT M,N:LOCATE X,Y,Z
```

If no move is possible (wall), then branch to line 580.

```
540 IF Z=1 THEN 580
```

If player hits a target block, then increment counter HT, and check for end of game.

```
550 IF Z=3 THEN HT=HT+1:FOR I=1 TO 20:
SOUND 0,I+50,12,15:NEXT I:IF HT=24 THE
N 800
```

Up until first hit, branch to line 560.

```
551 IF HT=0 THEN 560
```

Plot hits left of maze.

```
552 COLOR 3:IF HT/2=INT(HT/2) THEN PLO
T 1,23-HT
554 IF HT/2<>INT(HT/2) THEN PLOT 2,23-
HT
```

Set M and N to new location X and Y.

```
560 M=X:N=Y
```

Plot player's block.

```
580 COLOR 2:PLOT M,N
```

Increment time; plot time block on the right of the screen; check for end of game.

```
600 CT=CT+0.02:COLOR 2:PLOT 38,22-CT:P
LOT 39,22-CT:IF CT>21 THEN PLOT 38,0:P
LOT 39,0:GOTO 800
```

A little sound, and branch to beginning of game loop.

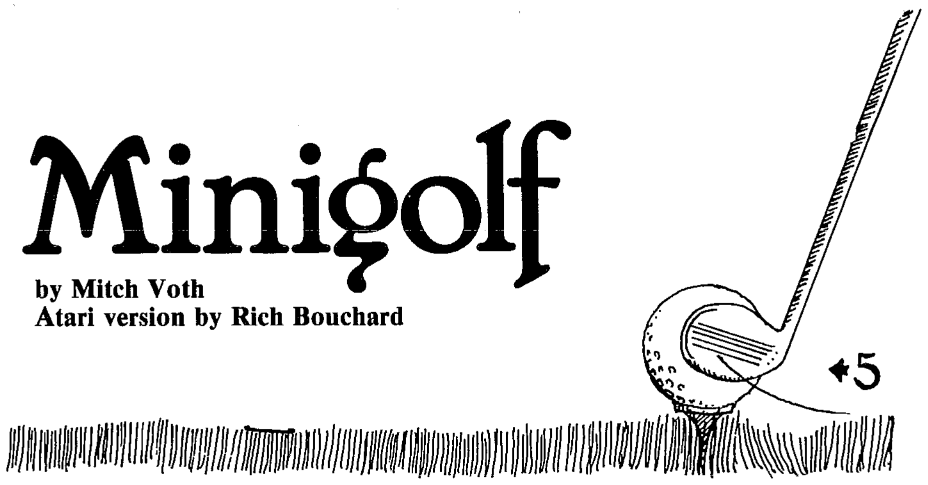
```
700 SOUND 0,2*M+2*N,10,4:GOTO 500
```

Game is over. Check for the fire button to start the next game.

```
800 IF STRIG(0)=0 THEN RUN
810 SOUND 0,RND(0)*100,10,4:GOTO 800
900 COLOR 3:X=INT(RND(0)*16)*2+5:Y=INT
(RND(0)*11)*2+1:LOCATE X,Y,Z
904 IF Z<>0 THEN 900
906 PLOT X,Y:RETURN
```

Minigolf

by Mitch Voth
Atari version by Rich Bouchard



***Minigolf* is an arcade/simulation game for an Atari with 16K RAM.**

Do you remember the golden days of youthful summers, when you and some friends would go to the miniature golf course? Do those days seem too far in the past? Here is a program that will let you recapture a bit of those days, without leaving the keyboard of your Atari. *Minigolf* is a graphic representation of a nine-hole miniature golf course with banked walls and even some sound. For more fun, the program accomodates up to ten players, although you can play alone if you so desire. In *Minigolf,* you control the angle of your putter by pressing the left and right arrows on the keyboard. Then, when you feel that the angle is correct, press a number from one to nine, indicating how hard you wish to strike the ball. One is a light tap, and nine is a hard hit. The player with the lowest number of strokes after nine holes.

Variables

A: General work variable.
B1, B2, C1, C2, D1, D2: Used in moving the ball.
HH: How hard the ball is hit.
HM: Horizontal movement.
M1, M2: Movement indicators.
NP: Number of players.
P: Par for each hole.
PT: Player's turn number.
S(i): Score for each player.
S: Number of strokes per hole.
VM: Vertical movement.
Z$(i): Player names.

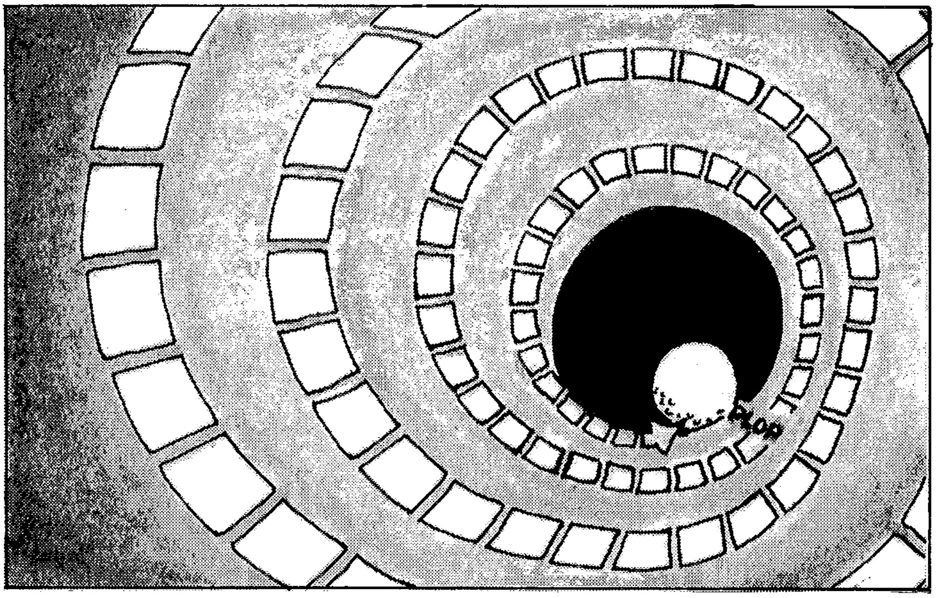
```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS         'Minigolf'         SS
SS    Author:  Mitch Voth     SS
SS Translator:  Rich Bouchard SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS

8 OPEN #1,4,0,"K:"

9 DIM Z$(100),A$(30),S(10)

10 GRAPHICS 0:PRINT "          ATARI
  9-HOLE":PRINT "          MINIATURE
GOLF"

20 FOR A=1 TO 100:Z$(A)=" ":NEXT A

25 FOR A=1 TO 10:S(A)=0:NEXT A

30 PRINT :PRINT :PRINT "NUMBER OF PLAY
ERS";:INPUT NP:IF NP>10 THEN 10

32 FOR A=1 TO NP:S(NP)=0:PRINT "PLAYER
```

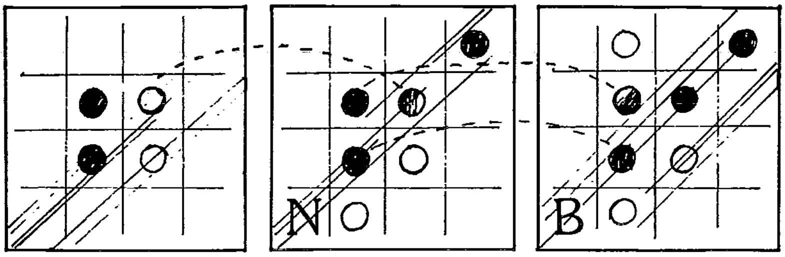
```
 #";A;:INPUT A$:IF LEN(A$)>10 THEN A$=
A$(1,10)
33 IF LEN(A$)=0 THEN A$=" "
34 Z$(A*10-LEN(A$)+1,A*10)=A$:NEXT A:G
RAPHICS 5:POKE 752,1
38 SETCOLOR 0,0,0:SETCOLOR 4,3,10:GOTO
 300
40 GRAPHICS 0:POKE 752,1:PRINT "
  SCORE:":PRINT :FOR A=1 TO NP:PRINT Z
$(A*10-9,A*10),S(A):NEXT A
42 PRINT :PRINT "HIT ANY KEY":GET #1,A
:GRAPHICS 5:POKE 752,1:SETCOLOR 0,0,0:
SETCOLOR 4,3,10:RETURN
50 GET #1,Z:IF Z<58 AND Z>48 THEN HH=(
Z-48)*11:HM=0:VM=0:COLOR A:PLOT C1,C2:
GOTO 200
60 T=0:IF Z<>43 THEN 70
62 CP=CP+1:IF CP=17 THEN CP=1
64 COLOR A:PLOT C1,C2:GOTO 100
70 IF Z<>42 THEN 50
72 CP=CP-1:IF CP=0 THEN CP=16
74 COLOR A:PLOT C1,C2:GOTO 100
90 COLOR 1:PLOT HP1,HP2:PLOT HP1+1,HP2
92 COLOR 3:PLOT B1,B2:LOCATE C1,C2,A:I
F A<>2 THEN COLOR 2:PLOT C1,C2:GOTO 50
94 COLOR 3:PLOT C1,C2:GOTO 50
100 GOTO CP+100
101 C1=B1:C2=B2+2:M1=0:M2=-1:GOTO 90
102 C1=B1+1:M1=-1:GOTO 90
103 C1=B1+2:C2=B2+2:M1=-2:GOTO 90
104 C2=B2+1:M2=-0.5:GOTO 90
105 C2=B2:M2=0:GOTO 90
106 C2=B2-1:M2=0.5:GOTO 90
107 C1=B1+2:C2=B2-2:M1=-2:M2=1:GOTO 90
108 C1=B1+1:M1=-1:GOTO 90
109 C1=B1:M1=0:GOTO 90
110 C1=B1-1:M1=1:GOTO 90
111 C1=B1-2:C2=B2-2:M1=2:M2=1:GOTO 90
112 C2=B2-1:M2=0.5:GOTO 90
113 C2=B2:M2=0:GOTO 90
114 C2=B2+1:M2=-0.5:GOTO 90
115 C1=B1-2:C2=B2+2:M1=2:M2=-1:GOTO 90
116 C1=B1-1:M1=1:GOTO 90
150 COLOR 3:PLOT HP1,HP2:PLOT HP1+1,HP
2:IF S>1 THEN 156
152 PRINT "A HOLE IN ONE!!!!!!":FOR A=
1 TO 5:FOR B=1 TO 59:SETCOLOR 4,B/4+1,
14:SOUND 0,B,0,10:NEXT B:NEXT A
154 SETCOLOR 4,3,10:SOUND 0,0,0,0:GOTO
 158
156 PRINT "THAT TOOK YOU ";S;" STROKES
":FOR A=1 TO 50:SOUND 0,A,10,10:NEXT A
:FOR A=49 TO 2 STEP -1
157 SOUND 0,A,10,10:NEXT A:SOUND 0,0,0
,0
158 FOR A=1 TO 700:NEXT A:PRINT CHR$(2
8);"
":PRINT CHR$(28);CHR$(28);:RETURN
200 D1=B1:D2=B2:COLOR 2:PLOT B1,B2:IF
(ABS(M1)=1 AND HM=1) OR ABS(M1)=2 THEN
 204
202 HM=1:GOTO 210
204 D1=B1+SGN(M1):HM=0
205 T=T+1:IF T>4 THEN RC=2:RETURN
210 SOUND 0,200,10,10:SOUND 0,0,0,0:LO
CATE D1,B2,A:IF A=0 THEN RC=0:RETURN
211 IF (D1<>HP1 AND D1<>HP1+1) OR D2<>
HP2 THEN IF A=1 THEN RC=0:RETURN
212 COLOR 3:PLOT D1,B2:B1=D1:LOCATE HP
1,HP2,A:IF A<>1 THEN S(PT)=S(PT)+S-P:R
C=1:GOSUB 150:RETURN
214 LOCATE HP1+1,HP2,A:IF A<>1 THEN S(
PT)=S(PT)+S-P:RC=1:GOSUB 150:RETURN
220 COLOR 2:PLOT B1,B2:IF (ABS(M2)=0.5
 AND VM=1) OR ABS(M2)=1 THEN D2=B2+SGN
(M2):VM=0:GOTO 230
222 VM=1
230 LOCATE B1,D2,A:IF A=0 THEN RC=0:RE
TURN
231 IF (D1<>HP1 AND D1<>HP1+1) OR D2<>
HP2 THEN IF A=1 THEN RC=0:RETURN
232 COLOR 3:PLOT B1,D2:B2=D2:LOCATE HP
1,HP2,A:IF A=3 THEN S(PT)=S(PT)+S-P:RC
=1:GOSUB 150:RETURN
234 LOCATE HP1+1,HP2,A:IF A=3 THEN S(P
T)=S(PT)+S-P:RC=1:GOSUB 150:RETURN
240 T=0:HH=HH-1:IF HH<0 THEN RC=2:RETU
RN
242 GOTO 200
300 COLOR 2:FOR A=3 TO 36:PLOT 27,A:DR
```

```
AWTO 45,A:NEXT A:PRINT "  HOLE # 1    P
AR 2"
302 P=2:FOR PT=1 TO NP:S=1:PRINT Z$(PT
*10-9,PT*10);"'S TURN":B1=INT(RND(0)*1
1+32):B2=31:CP=1
304 HP1=35:HP2=6
305 GOSUB 100
310 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 310
320 IF RC=1 THEN NEXT PT:GOTO 400
325 GOSUB 3000
330 IF D1=26 OR D1=46 THEN M1=-M1:GOSU
B 200:GOTO 310
332 M2=-M2:GOSUB 200:GOTO 310
400 GOSUB 40:COLOR 2:FOR A=13 TO 36:PL
OT 27,A:DRAWTO 45,A:NEXT A:FOR A=0 TO
11:PLOT 27+A,13-A:DRAWTO 70,13-A
402 NEXT A:PRINT "  HOLE # 2   PAR 2":
FOR PT=1 TO NP:S=1:PRINT Z$(PT*10-9,PT
*10);"'S TURN"
404 HP1=60:HP2=8:B1=INT(RND(0)*11+32):
B2=31
405 CP=1:GOSUB 100
420 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 420
430 IF RC=1 THEN NEXT PT:GOTO 500
435 GOSUB 3000
440 IF D2=1 OR D2=37 OR (D2=14 AND D1>
45) THEN M2=-M2:GOSUB 200:GOTO 420
442 IF D1<27 OR D1=46 OR D1=71 THEN M1
=-M1:GOSUB 200:GOTO 420
444 A=M1:M1=-M2*2:M2=-A/2:GOSUB 200:GO
TO 420
500 GOSUB 40:COLOR 2:FOR A=13 TO 36:PL
OT 17,A:DRAWTO 55,A:NEXT A:FOR A=0 TO
11:PLOT 17+A,13-A:DRAWTO 55-A,13-A
502 NEXT A:COLOR 1:PLOT 36,10:DRAWTO 3
6,36
510 PRINT "  HOLE # 3   PAR 2":FOR PT=
1 TO NP:PRINT Z$(PT*10-9,PT*10);"'S TU
RN":B1=INT(RND(0)*11+42)
512 S=1:B2=32:CP=1:HP1=25:HP2=32:GOSUB
 100
520 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 520
530 IF RC=1 THEN NEXT PT:GOTO 600
540 GOSUB 3000
550 IF D2=1 OR D2=37 THEN M2=-M2:GOSUB
 200:GOTO 520
552 IF D1=16 OR D1=36 OR D1=56 THEN M1
=-M1:GOSUB 200:GOTO 520
554 IF D1<36 THEN A=M1:M1=-M2*2:M2=-A/
2:GOSUB 200:GOTO 520
556 A=M1:M1=M2*2:M2=A/2:GOSUB 200:GOTO
 520
600 GOSUB 40:COLOR 2:FOR A=13 TO 24:PL
OT 21,A:DRAWTO 63,A:NEXT A:FOR A=25 TO
 36:PLOT 21,A:DRAWTO 35,A:NEXT A
602 FOR A=1 TO 8:PLOT 36+A,24+A:DRAWTO
 63-A,24+A:PLOT 21+A,13-A:DRAWTO 63-A,
13-A:NEXT A
604 COLOR 1:PLOT 36,25:DRAWTO 36,14:DR
AWTO 51,14:PLOT 51,14:DRAWTO 51,24
610 P=3:PRINT "  HOLE # 4   PAR 3":FOR
 PT=1 TO NP:PRINT Z$(PT*10-9,PT*10);"'
S TURN":B1=INT(RND(0)*8)+20
615 S=1:B2=32:CP=1:HP1=43:HP2=19:GOSUB
 100
620 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 620
630 IF RC=1 THEN NEXT PT:GOTO 700
640 GOSUB 3000
650 IF D2=4 OR D2=12 OR D2=37 OR (D2=3
3 AND D1>44 AND D1<55) OR (D2=14 AND D
1>36) THEN 654
652 GOTO 656
654 M2=-M2:GOSUB 200:GOTO 620
656 IF D1=20 OR D1=49 OR D1=51 OR D1=6
4 OR D1=36 THEN M1=-M1:GOSUB 200:GOTO
620
658 IF (D1>56 AND D2<13) OR (D2>24 AND
 D1<57) THEN A=M1:M1=M2*2:M2=A/2:GOSUB
 200:GOTO 620
659 A=M1:M1=-M2*2:M2=-A/2
660 GOSUB 200:GOTO 620
700 GOSUB 40:COLOR 2:FOR A=25 TO 36:PL
OT 41,A:DRAWTO 55,A:NEXT A:FOR A=1 TO
9:PLOT 33-A,25-A:DRAWTO 55,25-A
702 PLOT 24+A,11-A:DRAWTO 55-A,11-A:NE
XT A:FOR A=1 TO 5:PLOT 24,16-A:DRAWTO
```

```
55,16-A:NEXT A
704 COLOR 1:FOR A=1 TO 2:PLOT 55,25-A:
DRAWTO 45,15-A:NEXT A
710 PRINT "  HOLE # 5   PAR 2":P=2:FOR
 PT=1 TO NP:S=1:PRINT Z$(PT*10-9,PT*10
);"'S TURN":CP=1
715 B1=INT(RND(0)*8)+43:B2=33:HP1=33:H
P2=13:GOSUB 100
720 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 720
730 IF RC=1 THEN NEXT PT:GOTO 800
740 GOSUB 3000
750 IF D2=1 OR D2=37 OR (D2=25 AND D1<
41) THEN M2=-M2:GOSUB 200:GOTO 720
752 IF D1=23 OR D1=40 OR D1=56 THEN M1
=-M1:GOSUB 200:GOTO 720
754 IF D1<40 AND D2<11 THEN A=M1:M1=-M
2*2:M2=-A/2:GOSUB 200:GOTO 720
756 A=M1:M1=M2*2:M2=A/2:GOSUB 200:GOTO
 720
800 GOSUB 40:COLOR 2:FOR A=25 TO 36:PL
OT 31,A:DRAWTO 45,A:NEXT A:FOR A=1 TO
11:PLOT 31,25-A:DRAWTO 51+A,25-A
801 NEXT A
802 FOR A=2 TO 13:PLOT 31,A:DRAWTO 62,
A:NEXT A:COLOR 1:FOR A=1 TO 2:PLOT 30+
A,24:DRAWTO 39+A,15:NEXT A
804 PLOT 41,15:DRAWTO 41,10:PLOT 49,10
:DRAWTO 49,4
810 PRINT "  HOLE # 6   PAR 3":P=3:FOR
 PT=1 TO NP:S=1:PRINT Z$(PT*10-9,PT*10
);"'S TURN"
815 B1=INT(RND(0)*8)+34:B2=33:HP1=35:H
P2=13:CP=1:GOSUB 100
820 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 820
830 IF RC=1 THEN NEXT PT:GOTO 900
840 GOSUB 3000
850 IF D2=5 OR D2=1 OR D2=37 OR (D2=25
 AND D1>45) OR ((D2=10) AND (D1=41 OR
D1=49)) THEN M2=-M2:GOSUB 200:GOTO 820
852 IF D1=30 OR D1=46 OR D1=63 OR D1=4
9 OR (D1=41 AND D2<=16) THEN M1=-M1:GO
SUB 200:GOTO 820
854 A=M1:M1=-M2*2:M2=-A/2:GOSUB 200:GO
TO 820
900 GOSUB 40:COLOR 2:FOR A=1 TO 9:PLOT
 25-A,37-A:DRAWTO 54,37-A:PLOT 15+A,17
-A:DRAWTO 51-A,17-A:NEXT A
902 FOR A=21 TO 27:PLOT 16,A:DRAWTO 54
,A:NEXT A:FOR A=1 TO 4:PLOT 16,16+A:DR
AWTO 50+A,16+A:NEXT A
904 COLOR 1:PLOT 38,11:DRAWTO 50,23:PL
OT 38,16:DRAWTO 45,23:PLOT 26,27:DRAWT
O 38,27
906 PLOT 39,27:DRAWTO 39,36
910 PRINT "  HOLE # 7   PAR 3":P=3:FOR
 PT=1 TO NP:S=1:PRINT Z$(PT*10-9,PT*10
);"'S TURN"
915 B1=INT(RND(0)*8)+44:B2=33:HP1=33:H
P2=32:CP=1:GOSUB 100
920 IF RC=2 THEN S=S+1:CP=1:GOSUB 100:
GOTO 920
930 IF RC=1 THEN NEXT PT:GOTO 1000
940 GOSUB 3000
950 IF D1=15 OR D1=55 OR (D1=39 AND D2
>26) THEN M1=-M1:GOSUB 200:GOTO 920
952 IF D2=37 OR D2=7 OR D2=27 THEN M2=
-M2:GOSUB 200:GOTO 920
954 IF D1<25 AND D2<17 THEN A=M1:M1=-M
2*2:M2=-A/2:GOSUB 200:GOTO 920
956 A=M1:M1=M2*2:M2=A/2:GOSUB 200:GOTO
 920
1000 GOSUB 40:COLOR 2:FOR A=27 TO 36:P
LOT 29,A:DRAWTO 43,A:NEXT A:FOR A=1 TO
 8:PLOT 29-A,27-A:DRAWTO 43+A,27-A
1002 PLOT 21,19-A:DRAWTO 52,19-A:PLOT
21+A,11-A:DRAWTO 52-A,11-A:NEXT A
1004 COLOR 1:PLOT 30,11:DRAWTO 33,14:D
RAWTO 33,21:DRAWTO 36,24:PLOT 37,24:DR
AWTO 40,21:DRAWTO 40,14:DRAWTO 43,11
1010 PRINT "  HOLE # 8   PAR 3":P=3:FO
R PT=1 TO NP:S=1:PRINT Z$(PT*10-9,PT*1
0)
1015 B1=INT(RND(0)*8+32):B2=33:HP1=36:
HP2=17:CP=1:GOSUB 100
1020 IF RC=2 THEN S=S+1:CP=1:GOSUB 100
:GOTO 1020
1030 IF RC=1 THEN NEXT PT:GOTO 1100
1040 GOSUB 3000
```

```
1050 IF D2=37 OR D2=2 THEN M2=-M2:GOSU
B 200:GOTO 1020
1051 IF D1=33 OR D1=40 THEN IF D2=21 O
R D2=14 THEN 1054
1052 IF D1=20 OR D1=53 OR D1=33 OR D1=
40 OR D1=28 OR D1=44 THEN M1=-M1:GOSUB
 200:GOTO 1020
1054 IF (D1>43 AND D2>18) OR (D1<30 AN
D D2<11) OR (D1>36 AND D1<44) THEN A=M
1:M1=-M2*2:M2=-A/2:GOSUB 200:GOTO 1020
1056 A=M1:M1=M2*2:M2=A/2:GOSUB 200
1060 GOTO 1020
1100 GOSUB 40:COLOR 2:FOR A=27 TO 36:P
LOT 29,A:DRAWTO 43,A:NEXT A:FOR A=3 TO
 27:PLOT 20,A:DRAWTO 52,A:NEXT A
1102 COLOR 1:PLOT 34,10:DRAWTO 31,13:P
LOT 39,10:DRAWTO 42,13:PLOT 27,14:DRAW
TO 35,22:PLOT 38,22:DRAWTO 46,14
1110 PRINT "  HOLE # 9   PAR 4":P=4:FO
R PT=1 TO NP:S=1:PRINT Z$(PT*10-9,PT*1
0);"'S TURN"
1115 B1=INT(RND(0)*8+32):B2=33:HP1=36:
HP2=17:CP=1:GOSUB 100
1120 IF RC=2 THEN S=S+1:CP=1:GOSUB 100
:GOTO 1120
1130 IF RC=1 THEN NEXT PT:GOTO 2000
1140 GOSUB 3000
1150 IF D2=37 OR D2=2 OR D2=28 THEN M2
=-M2:GOSUB 200:GOTO 1120
1152 IF D1=19 OR D1=53 OR ((D1=28 OR D
1=44) AND D2>26) THEN M1=-M1:GOSUB 200
:GOTO 1120
1154 IF (D1>37 AND D2>13) OR (D1<36 AN
D D2<14) THEN A=M1:M1=-M2*2:M2=-A/2:GO
SUB 200:GOTO 1120
1156 A=M1:M1=M2*2:M2=A/2:GOSUB 200
1160 GOTO 1120
2000 GRAPHICS 0:PRINT "    FINAL SCORE
:":PRINT :PRINT "    PLAYER  SCORE":PR
INT "-----------------"
2005 FOR A=1 TO NP:PRINT Z$(A*10-9,A*1
0);" : ";S(A):NEXT A
2010 PRINT :PRINT "PLAY AGAIN (Y/N)";
2020 GOSUB 3000:A=PEEK(764):IF A<>43 A
ND A<>35 THEN 2020
2030 IF A=43 THEN RUN
2040 PRINT :PRINT :PRINT "GOOD BYE.":C
LOSE #1:END
3000 SOUND 0,50,10,10
3005 HH=HH-10:IF HH<2 AND HH>-6 THEN H
H=2
3010 SOUND 0,200,10,10
3020 SOUND 0,0,0,0:RETURN
9999 END
```





FLIP-IT

by Michael Prescott
Atari version by Alan J. Zett

***Flip-It* is an Atari program requiring 24K RAM.**

Flip-It is a computerized board game, in which you and the computer match wits trying to outflank and capture one another's pieces on an eight-by-eight board. The computer is a formidable opponent, and you won't find it a trivial matter to win.

The game begins with a square arrangement of four chips in the center of the board, two belonging to you and two belonging to the computer. You are allowed to choose your color and who will play first. When it is your turn, your object is to pick an unoccupied square such that putting one of your chips there will "outflank" one or more of the computer's chips. This means that the computer's chips would be sandwiched in between one of your existing chips and the new one you're playing, in a straight line. (You might think of it as your chips being bookends at each end of a row of your opponent's pieces.)

When you do this, all the computer's pieces which you have outflanked will become yours. This can happen in more than one direction, so that in any given turn you might capture pieces horizontally, vertically, and diagonally. This can result in a substantial shift in the relative number of chips in one turn, and the outcome of the game is rarely certain until the last few moves.

Use the joystick to move the cursor horizontally, vertically, or diagonally from its current location, and press the button to enter the move. The computer always checks to see that your move is legal, and allows no cheating.

During your turn, you may ask the computer to recommend your best move by pressing "B." If you see no move that you can make, you must press "N"; the computer will then check to see if you really have no possible move; if so, it will continue with its play. (Before pressing "B" or "N," you should move the cursor to an already occupied square.) If the computer has no move, it will pass after searching all the open squares. If neither of you has any possible move (which occasionally happens even before the board is filled), then the game is over and the player with the greater number of chips is the winner. You may also press "Q" to quit at any time.

You will soon find that winning a game involves more than just capturing as many pieces as you can on any given turn. Much more important to the eventual results will be the position of your chips on the board. Capturing edge and (especially) corner squares, and preventing the computer from doing the same will pay off in the long run, even if it means outflanking only one square when you could get more elsewhere on the board.

Variables
A(*): Decision-making array for computer's moves.
A$: Data manipulator.
B: Decision pointer.
C: Capture pointer.
CHIP: On-the-fly token.
COLR: Sampler variable.
COMMD: Key input variable
D: Loop variable.
IP: "I pass" flag.
OC: Token for computer's chip.
OPPONENT: Color of computer's chips.
P1: Number of chips captured.
P2: Number of chips on board.
P4: Sum of chips captured.
Q: Substitution variable.
RESTART: Re-entry line number.
SY: Number of your chips on board.
X,Y,Z: Loop variables.
X1,Y1: Point under consideration.
X2,X3: Check loop variables.
XX,YY: Plot variables.
XXX: Capture loop variable.
YC: Token for your chip.
YOUR: Color of your chips.
YP: "You pass" flag.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS         Atari BASIC        SS
SS          'Flip-It'         SS
SS  Author: Michael Prescott  SS
SS  Translator: Alan J. Zett  SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

Initialization.

```
100 GRAPHICS 0:DIM A(60),A$(120):Z=1:G
OSUB 6020:FOR X=1 TO LEN(A$) STEP 2:PD
L=0:B=0:FL=0
110 A(Z)=ASC(A$(X,X))-48+(ASC(A$(X+1,X
+1))-48)*10:Z=Z+1:NEXT X:GOSUB 5000:OP
EN #1,4,0,"K":PDL=0:B=0:F=0
120 POKE 764,255:B=C=D:P2=4:SY=2
```

Graphic introduction.

```
150 GRAPHICS 5:SETCOLOR 2,0,0:SETCOLOR
 4,0,0
160 FOR Y=0 TO 14 STEP 3:FOR D=0 TO 33
 STEP 0.2:SOUND 0,D,12,4:NEXT D:SOUND
0,0,0,4:IF (Y/2)=INT(Y/2) THEN 180
170 COLOR 2:SETCOLOR 1,Y,4:FOR X=12 TO
 39:PLOT X,0:DRAWTO X,38:PLOT 78-X,0:D
RAWTO 78-X,38:NEXT X:GOTO 190
180 COLOR 1:SETCOLOR 0,Y,4:FOR X=0 TO
19:PLOT 12,X:DRAWTO 66,X:PLOT 12,38-X:
DRAWTO 66,38-X:NEXT X
190 NEXT Y:SOUND 0,0,0,0:COLOR 0:FOR X
=1 TO 7:PLOT 7*X+11,0:DRAWTO 7*X+11,39
:PLOT 12,5*X-1:DRAWTO 67,5*X-1:NEXT X
```

Get color choice, and clear portion of memory that keeps track of game progress.

```
200 SETCOLOR 1,0,10:PRINT CHR$(125);"W
HICH COLOR (RED/ BLUE) ";:TRAP 200:INP
UT A$:IF A$(1,1)<>"R" THEN 210
205 YOUR=2:OPPONENT=3:GOTO 230
210 YOUR=3:OPPONENT=2
230 XX=14:YY=2:COLR=1
```

Prompt for type of game set-up, and choose who goes first.

```
240 PRINT CHR$(125);"WANT TO GO FIRST"
;:TRAP 240:INPUT A$
245 IF A$(1,1)="Y" THEN F=1
250 PRINT CHR$(125);"WANT TO SET UP YO
UR OWN GAME";:TRAP 250:INPUT A$
255 IF A$(1,1)="Y" THEN SETCOLOR 4,0,4
:SETCOLOR 0,1,2:GOTO 4000
260 TRAP 33333:GOSUB 6000:GOTO 600
300 ST=STICK(0)
310 IF ST=15 THEN ST=0
320 IF ST=14 AND PDL>7 THEN PDL=PDL-8
330 IF ST=13 AND PDL<56 THEN PDL=PDL+8
340 IF ST=11 AND PDL>0 THEN PDL=PDL-1
350 IF ST=10 AND PDL>8 THEN PDL=PDL-9
360 IF ST=9 AND PDL<56 THEN PDL=PDL+7
```

```
370 IF ST=7 AND PDL<63 THEN PDL=PDL+1
380 IF ST=6 AND PDL>7 THEN PDL=PDL-7
390 IF ST=5 AND PDL<55 THEN PDL=PDL+9
400 RETURN
```

Set up new game.

```
600 COLOR 3:X=45:GOSUB 1160:X=54:GOSUB
 1160:COLOR 2:X=44:GOSUB 1160:X=55:GOS
UB 1160
610 POKE 764,255
620 IF F=0 THEN 840
```

End if board is filled, otherwise update the chip-selector cursor.

```
630 IF P2=64 THEN 2010
632 GOSUB 7000
```

Get command, or update the chip-selector cursor.

```
640 IF PEEK(764)=255 THEN 800
645 CO=PEEK(764):POKE 764,255:IF CO<>4
7 AND CO<>35 AND CO<>21 THEN 800
650 IF CO=47 THEN 2012
652 IF CO=35 THEN 680
654 GOSUB 700:GOTO 800
```

No-possible-move command.

```
680 GOSUB 700:IF YP=1 THEN 840
682 SOUND 0,255,12,15:FOR X=1 TO 150:N
EXT X:SOUND 0,0,0,0:GOTO 800
```

Find best move.

```
700 FL=1:Q=YOUR:YOUR=OPPONENT:OPPONENT
=Q:GOSUB 2000:IF YP=1 THEN 710
702 FOR XXX=1 TO 20:IF (XXX-INT(XXX/2)
*2)=0 THEN COLOR 0
704 IF (XXX-INT(XXX/2)*2)<>0 THEN COLO
R 1
706 IF (XXX-INT(XXX/5)*5)<>0 THEN GOSU
B 1160
708 NEXT XXX
710 Q=YOUR:YOUR=OPPONENT:OPPONENT=Q:FL
=0:RETURN
800 GOSUB 300:IF A=PDL THEN 810
802 A=PDL:COLOR COLR:PLOT XX,YY:COLOR
0:XX=7*(A-INT(A/8)*8)+14:YY=INT(A/8)*5
+2
```

Position and display player's move.

```
804 LOCATE XX,YY-1,COLR:PLOT XX,YY
810 IF STRIG(0)=1 THEN 640
812 IF COLR<>1 THEN 640
820 P3=SY:COLOR YOUR:P4=0:D=0:FOR XXX=
1 TO 8:X=(A-INT(A/8)*8)+1+10*(INT(A/8)
+1):P1=0:GOSUB 980+XXX*20
822 P4=P4+P1:IF P1<>0 THEN D=D+1
830 NEXT XXX:IF D=0 THEN 640
832 SY=SY+P4-D+1:P2=P2+1
```

Do computer's move.

```
840 GOSUB 7000:GOSUB 2000:FOR XXX=1 TO
 42:IF (XXX-INT(XXX/2)*2)=0 THEN COLOR
 0
842 IF (XXX-INT(XXX/2)*2)<>0 THEN COLO
R 1
844 IF (XXX-INT(XXX/3)*3)=0 THEN GOSUB
 1160
846 NEXT XXX:COLOR OPPONENT
900 P2=P2+1:Q=YOUR:YOUR=OPPONENT:OPPON
ENT=Q:P4=0:D=0:FOR XXX=1 TO 8:X=A(B):P
1=0:GOSUB 980+20*XXX
905 IF P1<>0 THEN D=D+1:P4=P4+P1
910 NEXT XXX:SY=SY-P4+D
920 Q=YOUR:YOUR=OPPONENT:OPPONENT=Q:LO
CATE XX,YY-1,COLR:GOTO 630
```

Capture routine.

```
1000 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:Y1=Y1-5:IF Y1<0
THEN RETURN
1005 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1010 IF Z=OPPONENT THEN NEXT X2
1012 IF X2=8 THEN RETURN
```

```
1014 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-10:GOSUB 1160:NEXT X3:RETURN
1020 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:Y1=Y1+5:IF Y1>38
 THEN RETURN
1025 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1030 IF Z=OPPONENT THEN NEXT X2
1032 IF X2=8 THEN RETURN
1034 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+10:GOSUB 1160:NEXT X3:RETURN
1040 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1-7:IF X1<12
 THEN RETURN
1045 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1050 IF Z=OPPONENT THEN NEXT X2
1052 IF X2=8 THEN RETURN
1054 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-1:GOSUB 1160:NEXT X3:RETURN
1060 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1+7:IF X1>66
 THEN RETURN
1065 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1070 IF Z=OPPONENT THEN NEXT X2
1072 IF X2=8 THEN RETURN
1074 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+1:GOSUB 1160:NEXT X3:RETURN
1080 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1-7:Y1=Y1-5:
IF X1<12 OR Y1<0 THEN RETURN
1085 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1090 IF Z=OPPONENT THEN NEXT X2
1092 IF X2=8 THEN RETURN
1094 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-11:GOSUB 1160:NEXT X3:RETURN
1100 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1+7:Y1=Y1-5:
IF X1>66 OR Y1<0 THEN RETURN
1105 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1110 IF Z=OPPONENT THEN NEXT X2
1112 IF X2=8 THEN RETURN
1114 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-9:GOSUB 1160:NEXT X3:RETURN
1120 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1+7:Y1=Y1+5:
IF X1>66 OR Y1>38 THEN RETURN
1125 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1130 IF Z=OPPONENT THEN NEXT X2
1132 IF X2=8 THEN RETURN
1134 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+11:GOSUB 1160:NEXT X3:RETURN
1140 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1-7:Y1=Y1+5:
IF X1<12 OR Y1>38 THEN RETURN
1145 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1150 IF Z=OPPONENT THEN NEXT X2
1152 IF X2=8 THEN RETURN
1154 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+9:GOSUB 1160:NEXT X3:RETURN
```

Draw chip at selected location.

```
1160 X1=7*(X-INT(X/10)*10)+5:Y1=INT(X/
10)*5-5:FOR X1=X1 TO X1+5:PLOT X1,Y1:D
RAWTO X1,Y1+3:NEXT X1
1165 FOR Z=1 TO 6:POKE 53279,0:NEXT Z:
RETURN
```

Step through the best moves for the computer. If none are found, then pass or concede game.

```
2000 POKE 77,0:IF P2-SY=0 OR SY=0 THEN
 2010
2001 IF FL=0 THEN IP=0
2002 IF FL=1 THEN YP=0
2003 FOR B=1 TO 60:X=A(B):X1=7*(X-INT(
X/10)*10)+7:Y1=INT(X/10)*5-3:POKE 5327
9,0:POKE 53279,0:LOCATE X1,Y1,Z
2004 IF Z=1 THEN GOSUB 3000
2005 NEXT B:IF P2-SY<>0 AND SY<>0 AND
FL=0 AND YP=0 THEN 2008
2006 IF P2-SY<>0 AND SY<>0 AND FL=1 AN
D IP=0 THEN YP=1:RETURN
```

```
2007 GOTO 2010
2008 IP=1:FOR ST=170 TO 220:SOUND 0,ST
,10,10:NEXT ST:SOUND 0,0,0,0:GOTO 630
2010 FOR X=0 TO 200:SOUND 0,X,12,6:NEX
T X:SOUND 0,0,0,0:PRINT "NO POSSIBLE M
OVES LEFT!";:FOR X=1 TO 300:NEXT X
2012 POKE 752,1:IF P2-SY>SY THEN 2030
2014 IF P2-SY=SY THEN 2040
2020 PRINT :PRINT "YOU WON THIS ROUND
BY ";2*SY-P2;" CHIPS.";:GOTO 2050
2030 ? :? "I ONCE AGAIN PROVE MY SUPER
IORITY!";:FOR X=1 TO 300:NEXT X:? :? "
I WON BY ";P2-2*SY;" CHIPS.";
2032 GOTO 2050
2040 ? :? "IT'S A DRAW! I'LL BEAT YOU
NEXT TIME!!";
2050 FOR X=1 TO 20:POKE 53279,0:NEXT X
:FOR X=1 TO 300:NEXT X:PRINT :PRINT "W
ANT TO PLAY AGAIN?";:POKE 764,255
2060 CO=PEEK(764):IF CO=255 THEN 2060
2062 IF CO=43 THEN POKE 764,255:RUN
2064 POKE 764,255:? :? "OK. SEE YOU LA
TER!":PRINT :END
```

The computer has found an empty square; now check to see if it is a legal move.

```
3000 FOR C=1 TO 8:X1=7*(X-INT(X/10)*10
)+6:Y1=INT(X/10)*5-3:POKE 53279,0:POKE
 53279,0:GOSUB 2990+20*C:NEXT C:RETURN
```

Check legality of computer's moves (up-left, up-right, etc.). If it is a legal move, then jump out of loop, and do capture.

```
3010 FOR D=1 TO 7:Y1=Y1+5:IF Y1>38 THE
N RETURN
3012 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3014 IF Z=YOUR THEN NEXT D
3020 IF D=8 THEN RETURN
3022 POP :POP :POP :POP :POP :GOTO 515
0
3030 FOR D=1 TO 7:Y1=Y1-5:IF Y1<0 THEN
 RETURN
3032 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3034 IF Z=YOUR THEN NEXT D
3040 IF D=8 THEN RETURN
3042 POP :POP :POP :POP :POP :GOTO 515
0

3050 FOR D=1 TO 7:X1=X1-7:IF X1<12 THE
N RETURN
3052 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3054 IF Z=YOUR THEN NEXT D
3060 IF D=8 THEN RETURN
3062 POP :POP :POP :POP :POP :GOTO 515
0

3070 FOR D=1 TO 7:X1=X1+7:IF X1>66 THE
N RETURN
3072 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3074 IF Z=YOUR THEN NEXT D
3080 IF D=8 THEN RETURN
3082 POP :POP :POP :POP :POP :GOTO 515
0

3090 FOR D=1 TO 7:X1=X1+7:Y1=Y1+5:IF X
1>66 OR Y1>38 THEN RETURN
3092 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3094 IF Z=YOUR THEN NEXT D
3100 IF D=8 THEN RETURN
3102 POP :POP :POP :POP :POP :GOTO 515
0

3110 FOR D=1 TO 7:X1=X1-7:Y1=Y1+5:IF X
1<12 OR Y1>38 THEN RETURN
3112 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3114 IF Z=YOUR THEN NEXT D
3120 IF D=8 THEN RETURN
3122 POP :POP :POP :POP :POP :GOTO 515
0

3130 FOR D=1 TO 7:X1=X1-7:Y1=Y1-5:IF X
1<12 OR Y1<0 THEN RETURN
3132 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3134 IF Z=YOUR THEN NEXT D
3140 IF D=8 THEN RETURN
```

```
3142 POP :POP :POP :POP :POP :GOTO 515
0
3150 FOR D=1 TO 7:X1=X1+7:Y1=Y1-5:IF X
1>66 OR Y1<0 THEN RETURN
3152 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3154 IF Z=YOUR THEN NEXT D
3160 IF D=8 THEN RETURN
3162 POP :POP :POP :POP :POP :GOTO 515
0
```

Set up an old game.

```
4000 TRAP 33333:? CHR$(125);"HOW DO YO
U WANT TO SET IT UP?":? "1 - REGULAR,
2 - MANUAL":POKE 752,1:P2=0:SY=0
4010 GET #1,CO:IF CO<49 OR CO>50 THEN
4010
4020 POKE 764,255:IF CO=49 THEN 4250
4050 PRINT CHR$(125);"PRESS <ESC> WHEN
 FINISHED":PRINT "USE BUTTON TO SELECT
 SQUARE":POKE 752,1
4060 GOSUB 300:A=PDL:COLOR COLR:PLOT X
X,YY:COLOR 0:XX=7*(A-INT(A/8)*8)+15:YY
=INT(A/8)*5+2
4062 LOCATE XX,YY-1,COLR:PLOT XX,YY
4070 IF PEEK(764)=255 THEN 4080
4072 CO=PEEK(764):POKE 764,255:IF CO=2
8 THEN 4160
4080 IF STRIG(0)=1 THEN 4060
4082 PRINT CHR$(125);
4090 PRINT "WHICH COLOR (R/B) ";:GET #
1,ST:IF ST<>82 AND ST<>66 THEN 4082
4092 X=((A-INT(A/8)*8)+1)+(10*(INT(A/8
)+1))
4100 COLOR 3:IF ST=82 THEN COLOR 2
4102 GOSUB 4240:Y1=Y1+1:B=0:IF X1>18 T
HEN LOCATE X1-7,Y1,Z:B=B+(Z<>1)
4110 IF X1<60 THEN LOCATE X1+7,Y1,Z:B=
B+(Z<>1)
4112 IF Y1>4 THEN LOCATE X1,Y1-5,Z:B=B
+(Z<>1)
4114 IF Y1<34 THEN LOCATE X1,Y1+5,Z:B=
B+(Z<>1)
4120 IF X1>18 AND Y1>4 THEN LOCATE X1-
7,Y1-5,Z:B=B+(Z<>1)
4122 IF X1>18 AND Y1<34 THEN LOCATE X1
-7,Y1+5,Z:B=B+(Z<>1)
4130 IF X1<60 AND Y1>4 THEN LOCATE X1+
7,Y1-5,Z:B=B+(Z<>1)
4132 IF X1<60 AND Y1<34 THEN LOCATE X1
+7,Y1+5,Z:B=B+(Z<>1)
4134 IF B=0 THEN 4150
4140 GOSUB 1160:P2=P2+1:LOCATE XX,YY-1
,COLR:GOTO 4050
4150 IF P2=0 THEN 4140
4152 GOSUB 4220:PRINT "YOU'RE MAKING U
P THIS GAME! A PIECE   COULDN'T BE THE
RE!":FOR X=1 TO 300:NEXT X:GOTO 4050
4160 X=44:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4162 X=45:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4170 X=54:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4172 X=55:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4174 P2=0:FOR Y=1 TO 8:FOR A=1 TO 8
4180 X=10*Y+A:GOSUB 4240:Y1=Y1+1:X1=X1
+1:LOCATE X1,Y1,Z:IF Z=1 THEN 4190
4182 P2=P2+1:IF Z=YOUR THEN SY=SY+1
4190 NEXT A:NEXT Y:IF P2-SY=0 OR SY=0
THEN 4200
4192 IF P2=64 THEN 4210
4194 LOCATE XX,YY-1,Z:COLOR Z:GOSUB 60
00:GOTO 610
4200 GOSUB 4220:? CHR$(125);"YOU HAVE
TO GIVE ME A CHANCE!":P2=0:SY=0:FOR X=
1 TO 300:NEXT X:GOTO 4050
4210 GOSUB 4220:? CHR$(125);"YOU DIDN'
T LEAVE ANY ROOM!":FOR X=1 TO 300:NEXT
 X:GOTO 4050
4220 PRINT CHR$(125);:FOR X=1 TO 200:P
OKE 53279,0:NEXT X:RETURN
4230 GOSUB 4220:? CHR$(125);"THE CENTR
AL SQUARES MUST BE FILLED!":FOR X=1 TO
 300:NEXT X:GOTO 4050
4240 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:RETURN
```

```
4250 GOSUB 6000:P2=4:SY=2:GOTO 600
```

Print instructions for game.

```
5000 POSITION 2,12:PRINT "NEED INSTRUC
TIONS";:TRAP 5000:INPUT A$:IF A$(1,1)<
>"Y" THEN RETURN
5010 GRAPHICS 0:POKE 752,1:POSITION 14
,2:PRINT " FLIP-IT ":PRINT :PRINT "
 THE OBJECT OF THIS GAME IS TO"
5020 PRINT "COMPLETELY FILL THE BOARD
WITH AS     MANY PIECES OF YOUR COLOR
AS YOU CAN."
5030 PRINT "TO DO THIS YOU MUST OUTFLA
NK YOUR     OPPONENT'S PIECES AND FLIP
 THEM TO"
5040 PRINT "YOUR COLOR. OUTFLANKING CA
N OCCUR     HORIZONTALLY, VERTICALLY,
OR DIAG-"
5050 PRINT "ONOLLY. THE GAME ENDS WHEN
 THE BOARD  IS FULL OR WHEN BOTH PLAYE
RS CAN'T    MOVE. ":PRINT
5060 PRINT "WHOEVER HAS THE MOST PIECE
S WINS.":POSITION 5,23:PRINT "(PRESS A
NY KEY TO CONTINUE)";
5065 IF PEEK(764)=255 THEN 5065
5070 POKE 764,255:GRAPHICS 0:POKE 752,
1:POSITION 2,2:PRINT "YOU MAKE MOVES B
Y MOVING THE CURSOR"
5080 PRINT "WITH THE JOYSTICK TO THE S
QUARE YOU   DESIRE AND PRESS FIRE":PRI
NT
5090 PRINT "DURING THE GAME YOUR CAN A
LSO CHOOSE  ONE OF THE OPTIONS BY PRES
SING THE    KEY INDICATED:":PRINT
5100 PRINT "Q - TO QUIT THE GAME":PRIN
T :PRINT "N - NOT ABLE TO MOVE":PRINT
:PRINT "B - ASK FOR THE BEST MOVE"
5110 POSITION 7,23:PRINT "(PRESS ANY K
EY TO BEGIN)";
5130 IF PEEK(764)=255 THEN 5130
5140 POKE 764,255
5150 X=A(B):RETURN
```

Color set-up routine.

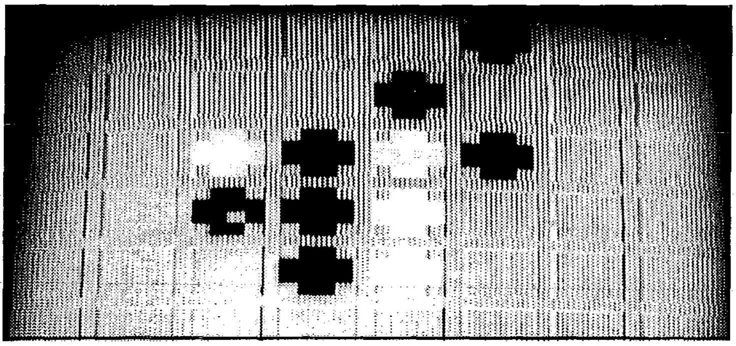
```
6000 TRAP 33333:GRAPHICS 5+48:SETCOLOR
 1,5,4:SETCOLOR 2,7,4:SETCOLOR 4,12,4:
SETCOLOR 0,0,0:RETURN
```

Data to set up square-choice priority for computer.

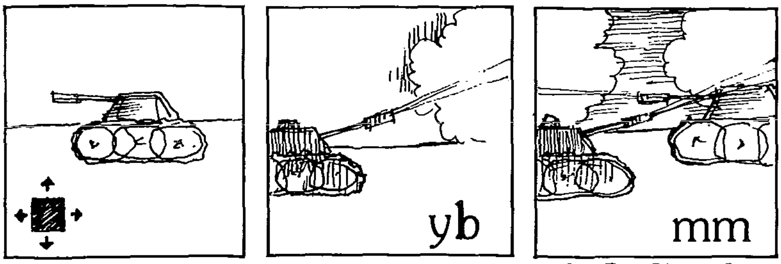
```
6020 A$="1181188831611383168638683363З
6664151148415854858435333464356546564 25
22474257547573262237326763767 2171"
6030 A$(LEN(A$)+1)="128217872878227227
77":RETURN
```

Chip-count update.

```
7000 COLOR 0:PLOT 0,40:DRAWTO 79,40:PL
OT 0,42:DRAWTO 79,42
7010 IF SY>0 THEN COLOR YOUR:PLOT 12,4
0:DRAWTO SY+11,40
7020 IF (P2-SY)>0 THEN COLOR OPPONENT:
PLOT 12,42:DRAWTO (P2-SY)+11,42
7030 RETURN
```



Battlefield



by Joe Humphrey
Atari version by Jon R. Voskuil

***Battlefield* is an Atari game requiring 16K RAM.**

"Battlefield" is a game of strategy for two players, in which the object is to overpower the opponent's forces and capture squares on the playing board. The players both start out with 40 squares on an eight-by-ten board, and alternate moving forces around the board to do battle with one another.

At the beginning of the first player's turn, new forces are added to each of the squares along the right and left edges of the board. The number of new forces that each player receives depends on how much territory he owns, how many edge squares he owns, and how crowded his edge squares are. No square may contain more than 99 forces.

At the same time, each player's Movement Allowance (M.A.) is increased, by an amount proportional to the territory he owns. The M.A. limits the amount of movement on each turn. For example, moving five forces through two squares, and then three other forces through six squares, uses up 5-times-2 plus 3-times-6, or 28, movement points. A player's turn ends when he has exhausted his Movement Allowance or when he presses "E". Unused M.A. is carried over to the next turn.

Forces are moved using the number keys and four directional keys. The board display will show a pair of "#" symbols bracketing one of the squares; these constitute the "cursor." The cursor can be moved up, left, right, or down by pressing I, J, K, or L (these keys form a diamond-shaped pattern for convenient right-hand use). To move forces, you position the cursor on a square occupied by your forces, type in the number of forces you want to move, and then use the directional keys to move them. To leave some forces in a given square, just type "0" or backspace to cancel the current number of forces being moved.

You can move your forces freely, in any direction, through your own territory. Once a group of forces crosses into enemy territory, however, they cannot move further during that turn. When a square is thus occupied by forces from both sides, a pair of letters is shown in that square rather than a number of forces. These letters show the relative number of forces in that square: The pair "YB" would show very unequal force strengths, while the pair "MM" would show equal forces from both sides. At the end of each turn, battles are fought to the death in all such jointly-occupied squares, and the surviving forces of the winner then remain in the squares.

These battles for individual squares are usually, but not always, won by the side with more forces in the square. Assume that the forces in a square are

15 of side A and 10 of side B. In the first round of the battle, there will be a 10/(15+10), or 0.4, probability that one of A's forces will be destroyed; at the same time there will be a 15/(15+10), or 0.6, probability that one of B's forces will be destroyed. At the end of the first round, then, the remaining forces could be 15:10, 14:10, 15:9, or 14:9. Assuming that the actual results are 15:9, the second round would be fought with new probabilities of 9/(15+9) and 15/(15+9), or 0.31 and 0.69 respectively. Rounds continue in this fashion until one player's forces are completely destroyed in that square.

If you should want to reduce the size of the "Battlefield" playing board, to shorten and simplify the game, it's easily done by changing line 70. NC is the number of columns and NR is the number of rows.

Variables

AR: Vertical tab location for Movement Allowance.
B$: Black's square marker.
BC$: Background character (CU$ or SP$).
BELL$: Beeps the speaker.
BL: Index to black player (=0).
BS$: Backspace character, CHR$(8).
CC: Cursor column.
CH$: A character.
CL$: Atari clear-screen character, CHR$(125).
COL: A column on the board.
CP: Controlling player in a square.
CP(i,j): Controlling player in each square of array.
CU$: Cursor character, "#".
D: Sound distortion number.
DC: Change in column CC (-1,0,+1).
DC(i): Change in column for each player.
DF: Digit flag; =1 if a digit was just typed.
DR: Change in row CR.
EC(i): End-of-line column for each player.
F: Boolean "false" value (=0).
FR: Vertical tab or PRINT@ position for number of forces.
HC: Half of NC.
IPL: Index to a player.
IV: Value for printing normal (=0) or inverse (=128) character.
LC(i): Label column for each player.
LP: Timing-loop counter.
MA(i): Movement allowance for each player.
MC: Maximum column index (=NC-1).
MF(i): Maximum number of incoming forces per square for each player.
MG: Maximum value of NG allowed.
MR: Maximum row index (=NR-1).
NB: Number of black forces in a square.
NC: Number of columns on board.
NF: Number of forces to add to a square.
NG: Number of forces in a moving group.
NR: Number of rows on board.
NS(i): Number of squares owned by each player.
NW: Number of white forces in a square.
PC: Player's name column.
PF: Previous value of DF.
PL: Index to current player.
PN$, PN1$, PN2$, PN$(i): Strings used for players' names.
PR: Player's-name row.
ROW: A row on the board.
S: Flag for sound generation.
SF: A square's number of forces.
SF(i,j), SF(i,j,k): Square's forces in each square, for each player.
SP$: Space character.
SR: Vertical printing position for squares owned.
T: Boolean "true" value (=1).
TF(i): Total forces for each player.
V: Sound volume value.
W$: White's square marker.
WH: Index to white player (=1).
WP$: Winning player's name.
X$: Temporary and utility string.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS         Atari  BASIC       SS
SS        'Battlefield'       SS
SS    Author:  Joe Humphrey   SS
SS Translator: Jon R. Voskuil SS
SS      Copyright (c) 1982    SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
10 GOTO 70
```

Subroutine to print a normal or inverse character.

```
20 FOR I=1 TO LEN(X$):PRINT CHR$(ASC(X
$(I,I))+IV);:NEXT I:RETURN
```

Subroutine to print the contents of position (COL, ROW) to the screen.

```
30 CP=CP(COL,ROW):IV=0:IF (BC$=SP$ AND
 CP=WH) OR (BC$=CU$ AND PL=WH) THEN IV
=128
32 IF S THEN SOUND 0,RND(1)*20+20,D,V
34 POSITION 4*COL+6,ROW+2:X$=BC$:GOSUB
 20:SOUND 0,0,0,0:POSITION 4*COL+3,ROW
+2:GOSUB 20
36 NB=SF(COL,ROW):NW=SF(COL+NC,ROW):SF
=NB+NW:IF SF(COL+(1-CP)*NC,ROW)=0 THEN
 40
38 IV=0:X$=CHR$(NB/SF*25+65):GOSUB 20:
IV=128:X$=CHR$(NW/SF*25+65):GOSUB 20:G
OTO 46
40 IV=0:IF CP=WH THEN IV=128
42 IF SF<10 THEN X$=" ":X$(2)=STR$(SF)
:GOSUB 20:GOTO 46
44 X$=STR$(SF):GOSUB 20
46 IV=0:RETURN
```

Main program control routine.

```
70 NC=8:NR=10
75 DIM CL$(1):CL$=CHR$(125)
78 POKE 752,1
80 GOSUB 1000
90 GOSUB 2000
100 GOSUB 3000
110 GOSUB 4000
120 GOSUB 5000
130 GOSUB 6000
140 GOSUB 7000
150 GOTO 8000
160 PRINT CL$:END
```

Print instructions.

```
1000 PRINT CL$;"BATTLEFIELD   *****
BY JOE HUMPHREY"
1010 PRINT :PRINT "  (ATARI TRANSLATIO
N BY JON VOSKUIL)"
1030 PRINT
1040 PRINT "    TWO PLAYERS, BLACK AND
 WHITE, ARE ON A ";NC;" BY ";NR;" BOAR
D.  A SQUARE OWNED"
1050 PRINT "BY BLACK IS SHOWN WITH THE
 NUMBER OF  FORCES IN THAT SQUARE AGAI
NST A BLACK"
1060 PRINT "BACKGROUND; A WHITE-OWNED
SQUARE HAS  A WHITE BACKGROUND."
1070 PRINT "    SQUARES CONTAINING BOT
H SIDES'    FORCES ARE SHOWN WITH LETT
ERS REPRE-"
1080 PRINT "SENTING THE RELATIVE FORCE
 SIZES.  AT THE END OF EACH TURN, BATT
LES ARE"
1090 PRINT "FOUGHT IN EACH OF THESE SQ
UARES, TO   DETERMINE WHO OWNS THEM.
THE FIRST"
1100 PRINT "PLAYER TO OWN ALL THE SQUA
RES WINS."
1110 PRINT "    AT THE BEGINNING OF EA
CH ROUND,   NEW FORCES ARE ADDED TO EA
CH SIDE,"
1120 PRINT "AND EACH SIDE'S MOVEMENT A
LLOWANCE IS INCREASED."
1130 RETURN
```

Initialize variables.

```
2000 BL=0:WH=1
2010 F=0:T=1
2020 MC=NC-1:MR=NR-1
```

```
2030 DIM DC(1),EC(1),LC(1),PC(1)
2040 DIM MA(1),MF(1),NS(1),TF(1),PN1$(
20),PN2$(20),WP$(15)
2050 DIM CP(MC,MR),SF(NC*2,MR)
2055 FOR I=0 TO 1:MA(I)=0:MF(I)=0:TF(I
)=0:NEXT I
2060 LC(BL)=4:DC(BL)=13:EC(BL)=21
2070 LC(WH)=LC(BL)+20:DC(WH)=DC(BL)+20
:EC(WH)=EC(BL)+19
2080 PR=15:FR=PR+2:SR=FR+2:AR=SR+2
2090 DIM BS$(1),SP$(1),CU$(1),BC$(1),X
$(20),CH$(1)
2100 SP$=" ":CU$="#"
2120 HC=INT(NC/2)
2130 FOR ROW=0 TO MR
2140 FOR COL=0 TO HC-1:CP(COL,ROW)=BL:
NEXT COL
2150 FOR COL=HC TO MC:CP(COL,ROW)=WH:N
EXT COL
2160 NEXT ROW
2170 NS(BL)=NR*HC:NS(WH)=NR*NC-NS(BL)
2175 FOR COL=0 TO NC*2:FOR ROW=0 TO MR
:SF(COL,ROW)=0:NEXT ROW:NEXT COL
2180 OPEN #2,4,0,"K:"
2190 RETURN
```

Input players' names.

```
3000 POKE 752,0:POSITION 2,22:PRINT "B
LACK PLAYER'S NAME: ";:INPUT PN1$
3010 IF PN1$="" THEN PN1$="BLACK"
3020 IF LEN(PN1$)>15 THEN PN1$=PN1$(1,
15)
3025 PC(BL)=9-LEN(PN1$)/2
3030 POSITION 2,23:PRINT "WHITE PLAYER
'S NAME: ";:INPUT PN2$
3040 IF PN2$="" THEN PN2$="WHITE"
3050 IF LEN(PN2$)>15 THEN PN2$=PN2$(1,
15)
3055 PC(WH)=29-LEN(PN2$)/2
3060 POKE 752,1:RETURN
```

Draw playing field, and display player data.

```
4000 BC$=SP$
4010 PRINT CL$
4020 FOR ROW=0 TO MR:FOR COL=0 TO MC
4030 GOSUB 30
4040 NEXT COL:NEXT ROW
4050 FOR IPL=BL TO WH
4070 POSITION PC(IPL),PR:IF IPL=BL THE
N PRINT PN1$;
4075 IF IPL=WH THEN PRINT PN2$;
4090 POSITION LC(IPL),FR:PRINT "FORCES
: ";
4095 POSITION DC(IPL),FR:PRINT TF(IPL)
;
4110 POSITION LC(IPL),SR:PRINT "SQUARE
S: ";
4115 POSITION DC(IPL),SR:PRINT NS(IPL)
;
4130 POSITION LC(IPL),AR:PRINT "MOVEME
NT: ";
4135 POSITION DC(IPL),AR:PRINT MA(IPL)
;
4140 NEXT IPL
4160 RETURN
```

Begin Black's turn. Add new forces to edge squares, and increase movement allowances.

```
5000 BC$=SP$
5005 S=1:D=10:V=8
5010 FOR IPL=BL TO WH
5020 MF(IPL)=INT(NS(IPL)/NR)+1
5030 NEXT IPL
5040 FOR ROW=0 TO MR:FOR COL=0 TO MC S
TEP MC
5050 CP=CP(COL,ROW):SF=SF(COL+CP*NC,RO
W)
5060 NF=MF(CP):IF SF+NF>99 THEN NF=99-
SF
5070 IF NF>0 THEN SF(COL+CP*NC,ROW)=SF
+NF:GOSUB 30
5080 TF(CP)=TF(CP)+NF:MA(CP)=MA(CP)+MF
(CP)
5090 NEXT COL:NEXT ROW
5100 POSITION DC(BL),FR:PRINT TF(BL);"
  ";:POSITION DC(WH),FR:PRINT TF(WH);"
  ";
```

```
5110 POSITION DC(BL),AR:PRINT MA(BL);"
  ";:POSITION DC(WH),AR:PRINT MA(WH);"
  ";
5120 PL=BL:CC=0
5125 S=0
5130 RETURN
```

Execute player's moves; move men until done or until his movement allowance is exhausted.

```
6000 CR=INT(NR/2)
6010 NG=0
6020 PF=F
6030 POSITION PC(PL),PR:IV=128:X$=PN1$
:IF PL=WH THEN X$=PN2$
6035 GOSUB 20
6040 BC$=CU$:COL=CC:ROW=CR:GOSUB 30
6050 GOSUB 11000
6060 MG=SF(CC+PL*NC,CR)
6070 IF MA(PL)=0 THEN 6120
6080 GOSUB 10000
6090 IF CH$="H" THEN 6030
6100 IF CH$=BS$ OR DF THEN 6050
6110 IF CH$<>"E" AND CH$<>"S" THEN 606
0
6120 BC$=SP$:COL=CC:ROW=CR:GOSUB 30
6130 NG=0:GOSUB 11000
6150 IV=0:POSITION PC(PL),PR:X$=PN1$:I
F PL=WH THEN X$=PN2$
6155 GOSUB 20
6160 IV=0:RETURN
```

Fight battles to the death in all squares that contain both players' forces.

```
7000 IF NS(PL)=0 THEN 7170
7005 S=1:D=4:V=12
7010 FOR ROW=0 TO MR:FOR COL=0 TO MC
7020 CP=CP(COL,ROW):IF CP=PL OR SF(COL
+PL*NC,ROW)=0 THEN 7160
7030 NS(CP)=NS(CP)-1
7040 NB=SF(COL+BL*NC,ROW):NW=SF(COL+WH
*NC,ROW)
7050 IF NW<>NB THEN CP(COL,ROW)=(NW>NB
)
7060 IF NW*NB=0 THEN 7130
7070 SF=NB+NW
7080 IF NB<SF*RND(1) THEN NB=NB-1:TF(B
L)=TF(BL)-1
7090 IF NW<SF*RND(1) THEN NW=NW-1:TF(W
H)=TF(WH)-1
7100 POSITION DC(BL),FR:PRINT TF(BL);"
  ";:POSITION DC(WH),FR:PRINT TF(WH);"
  ";
7110 SF(COL+BL*NC,ROW)=NB:SF(COL+WH*NC
,ROW)=NW
7120 GOSUB 30:GOTO 7050
7130 GOSUB 30
7140 CP=CP(COL,ROW):NS(CP)=NS(CP)+1
7150 POSITION DC(BL),SR:PRINT NS(BL);"
  ";:POSITION DC(WH),SR:PRINT NS(WH);"
  ";
7160 NEXT COL:NEXT ROW
7165 S=0
7170 RETURN
```

End of a turn. Check for end of game; if not, transfer control to player.

```
8000 IF NS(BL)*NS(WH)>0 THEN 8100
8020 WP$=PN1$:IF NS(BL)=0 THEN WP$=PN2
$
8025 POSITION 11,23:PRINT "
         ";
8030 FOR Z=1 TO 10:POSITION 2,21
8035 SOUND 0,50,10,10
8040 PRINT "
           ";:FOR ZZ=1 TO 10:NEXT ZZ
8045 SOUND 0,100,10,10
8050 POSITION 16-LEN(WP$)/2,21:PRINT W
P$;" WINS!!!";:FOR ZZ=1 TO 20:NEXT ZZ:
NEXT Z
8055 SOUND 0,0,0,0
8060 POKE 752,0:POSITION 14,23:PRINT "
ANOTHER GAME";:INPUT X$:IF X$="" THEN
8060
8070 IF X$(1,1)="N" THEN 160
```

```
8090 RUN
8100 IF PL=WH THEN 120
8110 PL=WH:CC=MC
8120 GOTO 130
```

Input and execute a command.

```
10000 GET #2,CH
10010 CH$=CHR$(CH)
10030 DF=(CH$>="0" AND CH$<="9"):DC=0:
DR=0
10040 IF DF AND  NOT PF THEN NG=0
10050 PF=DF
10060 IF DF THEN NG=10*NG+VAL(CH$):GOT
O 10160
10070 IF CH$=BS$ THEN NG=INT(NG/10):PF
=1:GOTO 10160
10080 IF CH$="E" THEN 10160
10090 IF CH$="H" THEN GOSUB 12000:GOSU
B 4000:GOTO 10160
10100 IF CH$="I" THEN DR=-1:GOTO 10150
10110 IF CH$="J" THEN DC=-1:GOTO 10150
10120 IF CH$="K" THEN DC=1:GOTO 10150
10130 IF CH$="M" THEN DR=1:GOTO 10150
10140 IF CH$="S" THEN NS(PL)=0:GOTO 10
160
10150 GOSUB 13000
10160 RETURN
```

Adjust the number of forces being moved, and update the screen display.

```
11000 IF CP(CC,CR)<>PL THEN NG=0
11010 IF NG>MG THEN NG=MG
11020 IF NG>MA(PL) THEN NG=MA(PL)
11030 POSITION 11,23
11040 IF NG=0 THEN PRINT "<TYPE H FOR
HELP>";:GOTO 11080
11050 PRINT "<MOVING ";NG;" FORCE";
11060 IF NG>1 THEN PRINT "S";
11070 PRINT ">";
11080 PRINT "    ";
11090 RETURN
```

```
12000 PRINT CL$:PRINT "COMMAND:","
 DESCRIPTION:"
12010 PRINT :PRINT "A NUMBER","NUMBER
OF FORCES TO MOVE."
12020 PRINT :PRINT "BACKSPC","DELETE L
AST DIGIT OF NUMBER."
12030 PRINT "   E","END TURN BEFORE MO
VEMENT=0."
12040 PRINT :PRINT "   H","PRINT THIS
COMMAND LIST."
12050 PRINT :PRINT "   I","MOVE SOME F
ORCES UP."
12060 PRINT :PRINT "   J","MOVE SOME F
ORCES LEFT."
12070 PRINT :PRINT "   K","MOVE SOME F
ORCES RIGHT."
12080 PRINT :PRINT "   M","MOVE SOME F
ORCES DOWN."
12090 PRINT :PRINT "   S","SURRENDER T
O OTHER SIDE."
12100 PRINT :PRINT :PRINT "  <HIT ANY
KEY TO RETURN TO GAME>";
12110 GET #2,X
12120 RETURN
```

Move cursor and forces, and update screen display.

```
13000 SF(CC+PL*NC,CR)=SF(CC+PL*NC,CR)-
NG
13010 BC$=SP$:COL=CC:ROW=CR:GOSUB 30
13020 CC=CC+DC:CC=CC+(CC<0)-(CC>MC)
13030 CR=CR+DR:CR=CR+(CR<0)-(CR>MR)
13040 FS=SF(CC+PL*NC,CR):IF FS+NG>99 T
HEN SF(COL+PL*NC,ROW)=SF(COL+PL*NC,ROW
)+NG-(99-FS):NG=99-FS:GOSUB 30
13050 SF(CC+PL*NC,CR)=FS+NG
13060 IF CC=COL AND CR=ROW THEN NG=0
13070 BC$=CU$:COL=CC:ROW=CR:GOSUB 30
13080 MA(PL)=MA(PL)-NG:GOSUB 11000
13090 POSITION DC(PL),AR:PRINT MA(PL);
"  ";
13120 RETURN
```

Solitaire

by Larry Williams
Atari version by Alan J. Zett

***Solitaire* is a graphic card game for a 24K Atari.**

Solitaire games are among the most popular card games. Although they lack the action of the arcade games, they can be every bit as addictive. What starts out as "one quick game" often turns into an evening of "one more game."

There are many variations of solitaire; however, the most widely known version is the game of Klondike. Both the lay-out and the rules of the game are simple. It is this simplicity that is perhaps its strength.

The Rules of Klondike

1. Use one deck of playing cards.
2. Deal twenty-eight cards into seven piles. The first pile on the left contains one card, the second contains two cards, and so forth.
3. Deal the top card of each pile face up.
4. Build on each pile in descending sequence and alternating colors. (Example: Any red ten may be played on any black jack.)
5. You are always entitled to have seven piles. If you clear away a pile leaving a space, you may play any king to the space.
6. The remaining twenty-four cards form the "stock." Cards are turned up one at a time from the stock onto the "waste" pile. The top card of the waste pile is always in play.
7. Any complete column of cards may be picked up as a unit and played on another pile. (Example: A column of cards consisting of the 9 of hearts, 8 of spades, and 7 of diamonds may be played as a unit on a 10 of spades.)
8. When a face-down card of a pile is uncovered, it is brought into play by turning it face up.
9. Whenever an ace comes into play, use it to start a "foundation" pile. Foundation piles are started to the side of the seven original piles of the tableau.
10. You build up on the foundation piles in suit and sequence.
11. The bottom card of any column of cards is eligible for play to its foundation.
12. Once a card is played to its foundation, it is out of play for the rest of the game.
13. The object of the game is to play each card to its foundation pile.

When the game program is RUN you will find the seven piles of the tableau laid out horizontally across the top of the playing surface. The first card in play is displayed in the lower right corner. A cursor is positioned at the bottom of the screen, and a summary of the commands available is displayed. When you start foundation piles they will appear at the right of the screen, above the waste pile.

The following single key commands are available:

LEFT ARROW: Moves the cursor to the left.
RIGHT ARROW: Moves the cursor to the right.
P: Picks up the card or column of cards above the cursor.
D: Drops cards which have been previously picked up at the cursor position, if the play is legal. You may always drop a card back where you picked it up.
N: Brings the next card from the stock into play by placing it on the top of the waste pile in the lower right of the screen. You are allowed only one pass through the stock.
F: Plays the card positioned above the cursor to its foundation pile, if the play is legal. (You need not type P to pick up a card that you wish to play to a foundation. Typing F alone will automatically move the card from its current location to its proper foundation.)
E: When you have no more legal plays, type E to end the game. The program will request confirmation of this command. Typing E will end your current game and begin a new one.

Variables

A$: Temporary string variable.
AZ: Loop variable.
C: Set equal to VA.
C(6,11): Array of card values in the tableau.
C$: Card values in letter form.
CHR: Related to ASCII value of current card value.
CPOS: Memory location of character data for current card value.
CU: Value of current cursor position (1-7).
D(51): The deck of cards.
F(4): Array of top cards on foundation piles.
G$(5): Graphics strings.
HF: Set to 1 if you have picked up cards in your hand, 0 if you do not.
I,IN,J: Counters.
IN(7): Array of counters pointing to the next open element in the tableau array and open deck.
NUM: Card and suit value.
OC: Old cursor position.
OD(23): Array for open deck.
PAUSE: Delay loop variable.
P(6,5): Array of face-down cards in tableau piles.
RN: Set to seed number for RND.
T: Temporary variable.
TX$(13): Array of text messages.
ST: Cursor position at which cards were last picked up.
SU: Suit of the card.
VA: Face value of the card.
VA$: Card values in letter form.
X: Loop variable.
X(7): X-coordinate values for tableau piles, waste pile (open deck), and foundations.
Y1,Y2,Y3,Y4: Y-coordinates for the four foundation piles.
Y(0-12): Y-coordinates for cards played to the tableau.
Y(13): Y-coordinate for the waste pile.
Z,ZA: Loop variables.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS         'Solitaire'        SS
SS   Author: Larry Williams   SS
SS  Translator: Alan J. Zett  SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
1 GRAPHICS 18:POKE 756,209
10 POSITION 5,3:? #6;"sOlItAiRe":POSIT
ION 3,5:? #6;"TRANSLATED BY":POSITION
4,7:? #6;"alan j zett":GOTO 1110
```

Subroutine to draw cards.

```
20 COLOR 1:FOR AZ=Y TO Y+30:PLOT X,AZ:
DRAWTO X+30,AZ:NEXT AZ
22 C=VA:ON SU GOTO 36,46,24,28
23 REM DIAMONDS
24 COLOR 0:FOR ZA=0 TO 9:FOR AZ=0 TO Z
```

```
A STEP 2:PLOT X+15+AZ,Y+ZA+6:PLOT X+15
-AZ,Y+ZA+6:NEXT AZ:NEXT ZA
26 FOR ZA=9 TO 0 STEP -1:FOR AZ=0 TO Z
A STEP 2:PLOT X+15+AZ,Y-ZA+25:PLOT X+1
5-AZ,Y-ZA+25:NEXT AZ:NEXT ZA:GOTO 56
27 REM HEARTS
28 COLOR 0:FOR ZA=0 TO 10:FOR AZ=0 TO
ZA STEP 2:PLOT X+15+AZ,Y-ZA+25:PLOT X+
15-AZ,Y-ZA+25:NEXT AZ:NEXT ZA
30 FOR ZA=11 TO 15:FOR AZ=0 TO 10 STEP
 2:PLOT X+15+AZ,Y-ZA+25:PLOT X+15-AZ,Y
-ZA+25:NEXT AZ:NEXT ZA
32 Z=-2:FOR ZA=16 TO 18:Z=Z+2:FOR AZ=Z
 TO 8-Z STEP 2:PLOT X+15+AZ,Y-ZA+25:PL
OT X+15-AZ,Y-ZA+25:NEXT AZ:NEXT ZA
34 GOTO 56
35 REM SPADES
36 COLOR 0:FOR AZ=0 TO 9:PLOT X+15+AZ,
Y+AZ+6:DRAWTO X+15-AZ,Y+AZ+6:NEXT AZ
38 FOR AZ=10 TO 12:PLOT X+15+9,Y+AZ+6:
DRAWTO X+15-9,Y+AZ+6:NEXT AZ
40 Z=-2:FOR ZA=13 TO 15:Z=Z+2:FOR AZ=Z
 TO 8-Z:PLOT X+15+AZ,Y+ZA+6:PLOT X+15-
AZ,Y+ZA+6:NEXT AZ:NEXT ZA
42 Z=0:FOR AZ=13 TO 19:Z=Z+1:PLOT X+15
+Z,Y+AZ+6:DRAWTO X+15-Z,Y+AZ+6:NEXT AZ
44 GOTO 56
45 REM CLUBS
46 COLOR 0:AZ=X+11:ZA=Y+6:GOSUB 52:AZ=
X+5:ZA=Y+13:GOSUB 52:AZ=X+17:GOSUB 52
48 Z=0:FOR AZ=14 TO 21:Z=Z+0.65:PLOT X
+15+Z,Y+AZ+5:DRAWTO X+15-Z,Y+AZ+5:NEXT
 AZ
50 FOR Z=X+14 TO X+16:PLOT Z,Y+13:DRAW
TO Z,Y+19:NEXT Z:GOTO 56
52 PLOT AZ+2,ZA:DRAWTO AZ+6,ZA:PLOT AZ
+2,ZA+7:DRAWTO AZ+6,ZA+7:PLOT AZ+1,ZA+
1:DRAWTO AZ+7,ZA+1
54 PLOT AZ+1,ZA+6:DRAWTO AZ+7,ZA+6:FOR
 Z=ZA+2 TO ZA+5:PLOT AZ,Z:DRAWTO AZ+8,
Z:NEXT Z:RETURN
56 CHR=ASC(C$(C,C)):IF CHR<96 THEN CHR
=CHR-32*(CHR>31)+64*(CHR<32)
58 CPOS=CHR*8+(PEEK(756)*256)
60 FOR AZ=0 TO 7:POKE (AZ*40)+MEM+(X/8
)+(Y*40),255-PEEK(CPOS+AZ):NEXT AZ:RET
URN
```

Subroutine to get SU and VA.

```
70 SU=INT(NUM/100)
80 VA=NUM-100*SU
90 RETURN
```

Subroutine for next card.

```
100 IF HF THEN GOSUB 1480:RETURN
110 IF IN>51 THEN 1490
120 OD(IN(7))=D(IN):IN=IN+1:X=X(7):Y=Y
(13):NUM=OD(IN(7)):GOSUB 70:GOSUB 20:I
N(7)=IN(7)+1
140 RETURN
```

Subroutine to draw cursor.

```
150 POKE 656,0:IF OC<7 THEN POKE 657,4
*OC+4:? " "
160 IF OC=7 THEN POKE 657,35:? " "
170 POKE 656,0:IF CU<7 THEN POKE 657,4
*CU+4:? "^"
180 IF CU=7 THEN POKE 657,35:? "^"
190 RETURN
```

Subroutine to move right.

```
200 CU=CU+1:IF CU>7 THEN CU=7
210 GOSUB 150
220 OC=CU
230 RETURN
```

Subroutine to move left.

```
240 CU=CU-1:IF CU<0 THEN CU=0
250 GOSUB 150
260 OC=CU
270 RETURN
```

Subroutine to pick up cards.

```
280 IF HF THEN GOSUB 1480:RETURN
290 ST=CU
300 IF IN(CU)=0 THEN GOSUB 1510:RETURN
310 IF CU=7 THEN NUM=OD(IN(7)-1):GOTO
330
320 NUM=C(CU,0)
330 HF=1
```

```
340 J=0:IF CU=7 THEN COLOR 1:FOR I=0 T
O 30:PLOT X(CU)+I,Y(13):DRAWTO X(CU)+I
,Y(13)+30:GOTO 370
350 IF P(CU,0)=0 THEN COLOR 0:FOR I=0
TO 30:PLOT X(CU)+I,Y(J):DRAWTO X(CU)+I
,Y(J)+30:GOTO 370
360 FOR I=0 TO 30 STEP 2:COLOR 0:PLOT
X(CU)+I,Y(J):DRAWTO X(CU)+I,Y(J)+30
365 IF I<30 THEN COLOR 1:PLOT X(CU)+I+
1,Y(J):DRAWTO X(CU)+I+1,Y(J)+30
370 NEXT I
380 IF CU=7 THEN GOSUB 150:RETURN
390 J=12*(IN(CU)-1)+32
395 COLOR 0:FOR I=0 TO 30:PLOT X(CU)+I
,Y(0)+31:DRAWTO X(CU)+I,Y(0)+J:NEXT I
400 RETURN
```

Main subroutine to drop card.

```
410 IF  NOT HF THEN GOSUB 1520:RETURN
420 IF CU=7 THEN GOSUB 590:RETURN
430 IF ST=CU THEN GOSUB 750:RETURN
440 IF IN(CU)=0 THEN GOSUB 630:RETURN
450 NUM=C(CU,IN(CU)-1)
460 GOSUB 70:TS=SU:TV=VA
470 IF ST=7 THEN NUM=OD(IN(7)-1):GOTO
490
480 NUM=C(ST,0)
490 GOSUB 70:IF ((TS=1) OR (TS=2)) AND
 ((SU=1) OR (SU=2)) THEN GOSUB 1530:RE
TURN
500 IF ((TS=3) OR (TS=4)) AND ((SU=3)
OR (SU=4)) THEN GOSUB 1540:RETURN
510 IF TV<>VA+1 THEN GOSUB 1550:RETURN
520 IF ST=7 THEN GOSUB 700:RETURN
530 FOR I=0 TO IN(ST)-1:NUM=C(ST,I):C(
CU,IN(CU))=NUM:GOSUB 70:X=X(CU):Y=Y(IN
(CU)):GOSUB 20:IN(CU)=IN(CU)+1
540 C(ST,I)=0:NEXT I:IN(ST)=0:HF=0
550 IF P(ST,0)=0 THEN RETURN
560 NUM=P(ST,0):GOSUB 70:X=X(ST):Y=Y(0
):GOSUB 20:C(ST,IN(ST))=NUM:IN(ST)=1
570 FOR I=0 TO 4:P(ST,I)=P(ST,I+1):NEX
T I:P(ST,5)=0
580 RETURN
```

Drop card back on open deck.

```
590 IF ST<>7 THEN GOSUB 1560:RETURN
600 NUM=OD(IN(7)-1):GOSUB 70:X=X(CU):Y
=Y(13):GOSUB 20:GOSUB 150
610 HF=0
620 RETURN
```

Drop king on open space.

```
630 IF ST=7 THEN NUM=OD(IN(7)-1):GOTO
650
640 NUM=C(ST,0)
650 GOSUB 70
660 IF VA<>13 THEN GOSUB 1570:RETURN
670 IF ST=7 THEN GOSUB 700:RETURN
680 GOSUB 530
690 RETURN
```

Drop card from open deck.

```
700 X=X(CU):Y=Y(IN(CU)):C(CU,IN(CU))=N
UM:GOSUB 20:IN(CU)=IN(CU)+1
710 IN(7)=IN(7)-1:OD(IN(7))=0:HF=0
720 IF IN(7)=0 THEN COLOR 0:X=X(7):Y=Y
(13):FOR I=0 TO 30:PLOT X+I,Y:DRAWTO X
+I,Y+30:NEXT I:RETURN
730 NUM=OD(IN(7)-1):GOSUB 70:X=X(7):Y=
Y(13):GOSUB 20
740 RETURN
```

Drop cards back where you got them.

```
750 FOR I=0 TO IN(CU)-1:NUM=C(CU,I):GO
SUB 70:X=X(CU):Y=Y(I):GOSUB 20:NEXT I
760 HF=0
770 RETURN
```

Play to foundations from open deck.

```
780 NUM=OD(IN(7)-1):GOSUB 70:FL=1
785 IF (F(SU)<>VA-1) AND (F(SU)=0) THE
N GOSUB 1580:RETURN
790 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB
 1550:RETURN
800 GOSUB 980
810 OD(IN(CU))=0
820 IF IN(CU)=0 THEN GOSUB 720:RETURN
830 GOSUB 730
```

```
835 GOSUB 150
840 RETURN
```

Play last card of column to foundation.

```
850 IF P(CU,0)=0 THEN COLOR 0:FOR I=0
TO 30:PLOT X(CU)+I,Y(0):DRAWTO X(CU)+I
,Y(0)+30:GOTO 875
860 FOR I=0 TO 30 STEP 2:COLOR 0:PLOT
X(CU)+I,Y(0):DRAWTO X(CU)+I,Y(0)+30
870 IF I<30 THEN COLOR 1:PLOT X(CU)+I+
1,Y(0):DRAWTO X(CU)+I+1,Y(0)+30
875 NEXT I:C(CU,0)=P(CU,0)
880 IF P(CU,0)=0 THEN RETURN
890 NUM=C(CU,0):X=X(CU):Y=Y(0):GOSUB 7
0:GOSUB 20
900 IN(CU)=1
910 FOR I=0 TO 4:P(CU,I)=P(CU,I+1):NEX
T I:P(CU,5)=0
920 RETURN
```

Main subroutine to play to foundations.

```
930 IF HF THEN RETURN
935 FL=0
940 IF IN(CU)=0 THEN GOSUB 1510:RETURN
950 IF CU=7 THEN GOSUB 780:RETURN
960 NUM=C(CU,IN(CU)-1):GOSUB 70
965 IF (F(SU)<>VA-1) AND (F(SU)=0) THE
N GOSUB 1580:RETURN
970 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB
 1550:RETURN
980 X=X(7)
990 IF SU=1 THEN Y=Y1
1000 IF SU=2 THEN Y=Y2
1010 IF SU=3 THEN Y=Y3
1020 IF SU=4 THEN Y=Y4
1030 GOSUB 20:F(SU)=VA
1040 IN(CU)=IN(CU)-1:IF FL THEN RETURN
1050 C(CU,IN(CU))=0
1060 IF IN(CU)=0 THEN GOSUB 850:RETURN
1070 X=X(CU):Y=Y(IN(CU)-1):NUM=C(CU,IN
(CU)-1):GOSUB 70:GOSUB 20
1080 COLOR 0
1090 FOR I=31 TO 45:PLOT X(CU),Y(IN(CU
)-1)+I:DRAWTO X(CU)+30,Y(IN(CU)-1)+I:N
EXT I
1100 RETURN
```

Initialize variables.

```
1110 CLR :DIM TX$(91),AZ$(7),C$(13),C(
6,11),P(6,5),D(51),OD(23),F(4),X(7),Y(
13),IN(7):OPEN #1,4,0,"K"
1115 TX$(91)="*":TX$(1,1)=" ":TX$(2)=T
X$(1):C$="A23456789TJQK":AZ$=TX$
1120 FOR I=1 TO 13:AZ$="       ":READ
AZ$:TX$(I*7-6,I*7)=AZ$:NEXT I
1150 FOR I=0 TO 6:FOR J=0 TO 5:C(I,J)=
0:P(I,J)=0:NEXT J:FOR J=6 TO 11:C(I,J)
=0:NEXT J:NEXT I
1160 FOR I=0 TO 23:OD(I)=0:NEXT I
1170 FOR I=0 TO 4:F(I)=0:NEXT I
1180 FOR I=0 TO 7:X(I)=21+I*32:Y(I)=I*
12:NEXT I:X(7)=269
1190 FOR I=8 TO 12:Y(I)=I*12:NEXT I
1200 Y1=0:Y2=32:Y3=64:Y4=96:Y(13)=128
1210 POKE 756,224
1220 IN=0:FOR I=1 TO 4:FOR J=1 TO 13:D
(IN)=100*I+J:IN=IN+1:NEXT J:NEXT I
```

Shuffle cards.

```
1230 FOR I=51 TO 0 STEP -1:X=INT(RND(0
)*(I+1)):T=D(X):D(X)=D(I):D(I)=T:NEXT
I
```

Deal to tableau.

```
1240 IN=0:FOR I=1 TO 6:FOR J=0 TO I-1:
P(I,J)=D(IN):IN=IN+1:NEXT J:NEXT I
1250 FOR I=0 TO 6:C(I,0)=D(IN):IN=IN+1
:NEXT I
```

Draw set-up.

```
1260 GRAPHICS 8:POKE 710,96:POKE 752,1
:COLOR 1:POKE 709,15:POKE 712,4:MEM=PE
EK(88)+PEEK(89)*256
1280 FOR X=254 TO 258 STEP 2:PLOT X,0:
DRAWTO X,160:NEXT X
1290 FOR I=0 TO 6:NUM=C(I,0):GOSUB 70:
X=X(I):Y=Y(0):GOSUB 20:NEXT I
1300 FOR I=0 TO 6:IN(I)=1:NEXT I:IN(7)
=0
1310 GOSUB 100:GOSUB 1460
1320 CU=0:DC=0:X=X(CU):GOSUB 150
```

Control program.

```
1330 POKE 764,255:GET #1,A:A=A-32*(A>9
0)
1340 IF A=42 THEN GOSUB 200:GOTO 1330
1350 IF A=43 THEN GOSUB 240:GOTO 1330
1360 IF A=78 THEN GOSUB 100:GOTO 1330
1370 IF A=80 THEN GOSUB 280:GOTO 1330
1380 IF A=68 THEN GOSUB 410:GOTO 1330
1390 IF A=70 THEN GOSUB 930:GOTO 1620
1400 IF A=69 THEN 1420
1410 GOTO 1330
```

Text messages.

```
1420 GOSUB 1600:? "DO YOU WANT TO END
THE GAME? (Y/N)";:GET #1,A
1430 IF A<>89 THEN GOSUB 1460:GOTO 133
0
1440 RUN
1450 GOSUB 1600:END
1460 GOSUB 1600:? "ARROWS MOVE, E=END
GAME F=FOUNDATION  N=NEXT CARD, P=PICK
 UP CARDS, D=DROP";
1470 RETURN
1480 GOSUB 1600:? "YOU'VE ALREADY PICK
ED UP A CARD";:GOTO 1610
1490 GOSUB 1600:? "THERE ARE NO MORE C
ARDS IN THE DECK   YOU MUST PLAY WITH
THE CARDS SHOWING";:GOTO 1610
1510 GOSUB 1600:? "THERE ARE NO CARDS
HERE TO PICK UP";:GOTO 1610
1520 GOSUB 1600:? "YOU DO NOT HAVE ANY
 CARDS TO DROP";:GOTO 1610
1530 GOSUB 1600:? "YOU CAN'T PLAY BLAC
K ON BLACK";:GOTO 1610
1540 GOSUB 1600:? "YOU CAN'T PLAY RED
ON RED";:GOTO 1610
1550 GOSUB 1600:? "YOU CAN'T DROP A";T
X$(VA*7-6,VA*7):? "ON TOP OF A";TX$(TV
*7-6,TV*7);:GOTO 1610
1560 GOSUB 1600:? "YOU CAN'T DROP CARD
S HERE";:GOTO 1610
1570 GOSUB 1600:? "YOU CAN ONLY DROP A
 KING HERE";:GOTO 1610
1580 GOSUB 1600:? "START YOUR FOUNDATI
ON WITH AN ACE";:GOTO 1610
```

Names of cards.

```
1590 DATA N ACE, TWO, THREE, FOUR, FIV
E, SIX, SEVEN,N EIGHT, NINE, TEN, JACK
, QUEEN, KING
```

Clear bottom of screen.

```
1600 POKE 656,2:POKE 657,2:? CHR$(156)
;CHR$(156);CHR$(253);:RETURN
```

Pause subroutine.

```
1610 FOR PAUSE=1 TO 300:NEXT PAUSE:GOS
UB 1460:RETURN
```

Check for game won.

```
1620 IF F(1)<13 OR F(2)<13 OR F(3)<13
OR F(4)<13 THEN GOTO 1330
1630 GOSUB 1600:? " YOU WIN!!  CARE TO
 PLAY AGAIN? (Y/N)";:GET #1,A:IF A<>78
 THEN RUN
1640 GRAPHICS 0:CLR
```

Gambler

by Randy Hawkins   Atari version by Rich Bouchard

***Gambler* is a game program for up to four players requiring a 32K Atari.**

Gambler is more than a dozen different games of chance. You and as many as three of your friends compete in a contest to see who can parlay a $100 stake into $1000 first. The computer would not be content to just keep score and watch all this money change hands so it becomes a "player" with just the same chances of winning and losing as you have.

On each player's turn, he is first given the opportunity to buy as many as three lottery tickets. A lottery ticket will return $50 to the owner if it matches the winning numbers in the lottery. After purchasing any tickets desired, the wheel of fortune is shown. A block of light dances randomly around the board stopping momentarily beside the name of the many games available. The player touches any key, and the dot slowly settles into one category — the game in which the player will participate.

There are many types of games available in *Gambler*. Some are very simple; you may be instructed by a mysterious fortune teller to give or receive amounts of money to or from other players or the bank. The names of some of these categories are somewhat deceptive so pay close attention as the message slowly scrolls across the screen.

There are also several betting games involving dice. You can win or lose money by betting on the outcome of the two dice rolls: will the total be even, will your roll be highest, or will it be the lucky number 7? There is even a form of "dice poker" with the best hand taking the pot.

Gambler even has its own horse racing track so that all the players can bet on the outcome and try to win the purse. Lotteries are also held once in a while to pay off the winning lottery tickets. Big money is available whenever a sweepstakes is held. The correct bet on the outcome of the roll of six dice can bring the winner up to $450.

Gambler is loosely based on a board game with many extras added. Because the computer always plays, you can practice with it. And since as many as four humans can play along with the computer, it makes a great game for parties or when you have a large crowd eager to play with the computer.

The winner is the first player to go over the $1000 mark. In case of a tie, the game continues until someone gets ahead. Should you go broke, the bank will advance you $100 but will charge you with one IOU. As soon as you can, you must pay the IOU back at a price of $110. These transactions will take place only on your turn.

So the time has come. Type in this program, gather three of your friends around the video screen, and see who is the champion gambler in your household.

Variables

A$: Contains name of game read from the DATA statements in lines 390-410.
A1$, A2$: Graphics strings used as borders in game selection screen.
B: ASCII code of user response.
D(6): Utility variables for six rolls of dice, six horse race positions, etc.
D$: Contains the pictures of the six dice.
GN: Game number selected in selection routine.
H$: Contains picture of the horses used in horse race subroutine.
HM: Contains highest money total when searching for winner.
HN: Number of player with highest money total.
I, J, K, L, S: Utility FOR-NEXT counters.
IO(X): Contains number of IOUs held by player X.
IP: Used as a loop variable to signal which player's turn it is.
JJ: Upper boundary on the FOR-NEXT loop that increases gradually to slow movement of the dot in game selection function.
L(X): Number of lottery tickets held by player X.
L$: Lottery ticket string.
M(I): Contains amount of money help by player I.
M$: Used to hold the message printed out during various functions.
N: Contains the number of players.
N$: Contains players' names.
PB: Contains the number of IOUs which can presently be paid off.
R$: Used as a utility string in construction of graphics.
T$, T1$: Used to generate an individual lottery ticket.
TI: Used as a timing delay.
X, Y, X1, Y1: Used to hold present and previous print location for dot in game selection routine.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS         'Gambler'          SS
SS   Author:  Randy Hawkins   SS
SS Translator:  Rich Bouchard SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

Program Initialization. Line 20 creates A1$ and A2$, graphics strings used in the borders of the game selection screen. Subroutine 420 does further initialization of graphics. Lines 30 and 40 set the number of players, N. Line 50 determines players' names and initializes money, M(I), numbers of IOUs, IO(I), and lottery tickets, L(I). If desired, instructions are displayed after prompting in line 50.

```
1 GRAPHICS 0
12 POKE 82,1:OPEN #2,4,0,"K:"
15 DIM N$(50),Z$(40),L$(50),M(6),IO(6)
,L(6),T$(3),T1$(3),A1$(40),A2$(40),A$(
30),D$(102),R$(4),O$(39),S$(6),B$(1)
16 DIM D(6),H$(16),M$(140),P(6)
17 FOR I=1 TO 6:L(I)=0:NEXT I
18 FOR T=1 TO 50:N$(T,T)=" ":NEXT T:L$
=N$:O$=N$(1,39)
20 A1$="_                   ":A1$(3,3)=
CHR$(25):A1$(LEN(A1$)+1)=A1$:A2$="____
________________________________"
22 A1$(LEN(A1$)+1)="_":GOSUB 420
30 PRINT "How many players (up to 4) ?
 ";
40 GET #2,N:N=N-48:IF N<1 OR N>4 THEN
40
42 PRINT N
50 FOR I=1 TO N:? "Who is player #";I;
```

```
:INPUT Z$:N$(I*10-9,I*10)=Z$:M(I)=100:
IO(I)=0
52 L(I)=0:NEXT I:N$(I*10-9,I*10)="ATAR
I     ":M(I)=100:IO(I)=0:N=N+1:PRINT "
Do you need to see instructions?";
60 GET #2,B:IF CHR$(B)="Y" THEN GOSUB
1370
```

Game loop. Loop, using variable IP, cycles through each player's turn. Present money totals are shown in line 80. Broke players are handled by lines 90-120. Lottery tickets are bought in lines 130-150. Game number is selected in subroutine 290, and line 180 transfers control to the appropriate routine.

```
70 FOR IP=1 TO N
80 GRAPHICS 0:PRINT "Present bankrolls
                IOUs":? "rrrrrrrrrrrr
rrrrrrrrrrrrrrrrrrrrrrrrrrr";
81 REM Line 80: Second string=
                39 <CTRL> R's
82 FOR J=1 TO N:PRINT N$(J*10-9,J*10);
"            $";M(J):POSITION 35,J+1:? I
O(J):NEXT J:? :GOSUB 200
84 PRINT "It is ";N$(IP*10-9,IP*10);"'
s turn.":? :IF M(IP)>=0 THEN 100
90 IO(IP)=IO(IP)+1:M(IP)=M(IP)+100:IF
M(IP)<=0 THEN 90
92 PRINT "You must borrow money to con
tinue to   play and pay it back to the
 bank ....  plus interest!"
94 FOR TI=1 TO 800:NEXT TI:GOTO 80
100 IF IO(IP)<=0 OR M(IP)<110 THEN 130
110 PB=INT(M(IP)/110):IF PB>IO(IP) THE
N PB=IO(IP)
120 IO(IP)=IO(IP)-PB:M(IP)=M(IP)-110*P
B:PRINT "You can pay back ";PB;" IOU n
ote(s)":? "at 110 dollars each."
122 FOR TI=1 TO 800:NEXT TI
130 IF L(IP)=3 THEN 160
132 PRINT "Would you like to buy a lot
tery        ticket for $10 ? ";
140 IF IP=N THEN PRINT "Y":GOTO 154
150 GET #2,B:IF CHR$(B)="N" THEN 160
152 IF CHR$(B)<>"Y" THEN 150
154 ? CHR$(253):GOSUB 260:L(IP)=L(IP)+
1:L$(IP*9+L(IP)*3-11,IP*9+L(IP)*3-8)=T
$:M(IP)=M(IP)-10
156 FOR TI=1 TO 400:NEXT TI:GOTO 80
160 ? :POSITION 4,20:? "Hit any key to
 continue";
170 GET #2,B:GOSUB 290
180 ON GN GOSUB 1020,810,460,1130,1170
,1190,460,1200,1230,1240,460,750,1300,
940,1020,710,810,1310,1320,750,460
182 IF GN>21 THEN ON GN-21 GOSUB 1330,
710,810,1340,1350,1330,810,460,940,131
0,710,750,1360,810,710
190 IP=IP+1:IF IP>N THEN 70
192 GOTO 80
```

Called in game loop to check for winner. The highest money total is found in line 200. If greater than $1000, then winner is notified; otherwise, game continues. Option of playing a new game is in lines 240:250.

```
200 HM=M(1):HN=1:FOR J=2 TO N:IF M(J)>
HM THEN HM=M(J):HN=J:NEXT J:GOTO 210
202 NEXT J
210 IF HM<1000 THEN RETURN
220 K=0:FOR J=1 TO N:IF M(J)=HM THEN K
=K+1
222 NEXT J
230 IF K>1 THEN PRINT "Since there's a
 tie, keep going":? :RETURN
240 ? :? N$(HN*10-9,HN*10);" wins this
 game with ":? HM;" dollars!"
242 FOR T=1 TO 100:SOUND 0,T,10,10:FOR
 TI=1 TO 5:NEXT TI:SOUND 0,255-T,10,10
:FOR TI=1 TO 5:NEXT TI:NEXT T
244 SOUND 0,0,0,0:SOUND 1,0,0,0
248 ? :? "Would you like to play again
 ?";
250 GET #2,B:IF CHR$(B)="Y" THEN RUN
252 END
```

Lottery subroutine. Selects two-digit lottery ticket that is different from all other lottery tickets issued. Displays results in line 280, and returns.

```
260 T$=" - ":T$(1,1)=CHR$(49+INT(RND(0
)*5)):T1$=CHR$(49+INT(RND(0)*6)):IF T1
$<=T$ THEN 260
```

```
262 T$(3,3)=T1$:FOR I=1 TO N:IF L(I)=0
 THEN 280
270 FOR J=1 TO L(I):IF T$=L$(I*9+J*3-1
1,I*9+J*3-9) THEN 260
272 NEXT J
280 NEXT I:PRINT "You receive ticket n
umber ";T$:RETURN
```

Game selector. Prints border, using A1$ and A2$. Reads names of games from DATA statements in lines 390-410. Lines 320-350 wait for players to touch any key. Lines 360-380 settle on one game, set the game number GN, and return.

```
290 RESTORE :GRAPHICS 0:PRINT A2$;:FOR
 J=1 TO 18:PRINT A1$;:NEXT J:PRINT A2$
;
300 FOR X=1 TO 18:READ A$:POSITION 4,X
:? A$:READ A$:POSITION 23,X:? A$:NEXT
X
310 X=1:X1=2:POKE 752,1:POKE 764,255:P
OSITION 2,22:? "Touch any key, ";N$(IP
*10-9,IP*10);
320 Y=X:Y1=X1:POSITION X1,X:? CHR$(160
);:X=INT(RND(0)*18)+1:X1=INT(RND(0)*2)
*19+2
322 FOR S=X*2+X1/21+50 TO X*2+X1/21+42
 STEP -4:SOUND 0,S,10,10:NEXT S:SOUND
0,0,0,0
330 IF IP=N AND RND(0)<0.03334 THEN 36
0
340 IF IP<>N THEN B=PEEK(764):IF B<>25
5 THEN 360
350 POSITION Y1,Y:? " ";:GOTO 320
360 POKE 764,255:JJ=25:FOR I=1 TO 6+IN
T(RND(0)*4):FOR J=1 TO JJ:NEXT J:JJ=JJ
+25:POSITION Y1,Y:? " ";:Y=X:Y1=X1
361 SOUND 0,X*2+X1/21+30,10,10:SOUND 0
,X*2+X1/21+40,10,10:SOUND 0,0,0,0
362 POSITION X1,X:? CHR$(160);:IF X1=2
 THEN X1=21:GOTO 370
364 X1=2:X=X+1:IF X=19 THEN X=1
370 NEXT I
380 FOR TI=1 TO 150:NEXT TI:GN=X*2+INT
(X1/13)-2:POSITION Y1,Y:? " ";:FOR I=1
 TO 5:POSITION Y1,Y:? CHR$(160);
381 FOR S=10 TO 120 STEP 10:SOUND 0,S,
10,10:NEXT S:FOR S=110 TO 0 STEP -10:S
OUND 0,S,10,10:NEXT S
382 FOR TI=1 TO 30:NEXT TI:POSITION Y1
,Y:? " ";:FOR TI=1 TO 30:NEXT TI:NEXT
I:POKE 752,0:RETURN
390 DATA POKER PARTY,F O R T U N E !,S
WEEPSTAKES!,LOVE A NEIGHBOR,EZ COME EZ
 GO,WIN A FEW
392 DATA SWEEPSTAKES!,UNLUCKY SEVEN,LO
SE A FEW
400 DATA EVEN STEVEN,SWEEPSTAKES!,DAIL
Y DOUBLE,POT LUCK,HIGH ROLLER,POKER PA
RTY,LOTTERY,F O R T U N E !
402 DATA JACKPOT,$100 BONUS,OFF TO THE
 RACES,SWEEPSTAKES!,LOSE THIS TURN
410 DATA LOTTERY,F O R T U N E !,TAX T
IME,BONANZA,LOSE THIS TURN,F O R T U N
 E !,SWEEPSTAKES!,HIGH ROLLER
412 DATA JACKPOT,LOTTERY,HORSE RACE,MA
D MONEY,F O R T U N E !,LOTTERY
```

Graphics initialization. Set up graphics variable D$, which is pictures of the six dice.

```
420 D$="   =+++ t =+++     t=+++   =++
+t     t=+++ t =+++t  t t=+++   =+++t t
t t=+++ t =+++t tt t=+++t t=+++t t"
421 REM Line 420: Non-space characters
  in string are control characters.
430 REM Above: T=<CTRL> T
432 REM d=<ESC> <CTRL> =
434 REM l=<ESC> <CTRL> +
450 RETURN
```

Sweepstakes routine. Lines 460-480 set up the graphics display. Line 490 offers optional instructions, which are in lines 510-530. Lines 540-600 record each player's sweepstakes bet in S$(J,J). Line 620 rolls six dice, displays them, and records them in D(X). The highest number rolled is stored in HN. Line 630 checks for pairs, 640-650 for a straight, 660 for a split, and 670 for single spots.

```
460 GRAPHICS 0:? "        S W E E P S
T A K E S !":? :? "    qrrrrGrrrrrwrr
rrrHrrrrre"
```

```
461 REM Line 460: Lower-case letters
are control characters.
464 ? "    A            C            E":
? "    !            !            !"
466 REM Line 464: Type [SHIFT]=
instead of !
470 ? :? :? :? "              !
!          !":? "P=Pair    B
  D          F"
471 REM Line 470: Type [SHIFT]=
instead of !
472 ? "S=StraightzrrrrIrrrrxrrrrJrr
rrrc"
473 REM Line 472: Lower-case letters
are control characters.
480 FOR X=1 TO 6:POSITION X*6-2,5:? D$
(X*17-16,X*17);:NEXT X
490 POSITION 1,12:? "Do you need Sweep
stakes instructions ?";:POKE 764,255
500 GET #2,B
510 IF CHR$(B)="N" THEN 540
512 POSITION 1,12:? "In Sweepstakes ev
eryone antes $10. Theneach player bets
 on the outcome of the"
514 ? "roll of 6 dice.  There are thre
e types of bets.":GOSUB 1470
520 POSITION 1,12:? "One way is to bet
 on where the highest number rolled wi
ll appear.  If the high";
522 ? "number appears in A,B,C,D,E or
F you   win $300.  Letters G,H,I and J
 cover   ";
524 ? "two spots.  If you bet G, you c
over    spots A and C and can win $150
."
530 GOSUB 1470:POSITION 1,12:? "Anothe
r bet is <<P>> for pairs.  If twodice
next to one another match, you win";
532 ? "$200.  The third bet is <<S>> f
or a    straight.  If 3 consecutive di
ce appear";
535 ? "in numerical order, you win $45
0.      ";O$:GOSUB 1470
540 FOR I=12 TO 18:POSITION 1,I:? O$;:
NEXT I:POSITION 30,12:? "A-F $300":POS
ITION 30,13:? "G-J $150"
542 POSITION 32,14:? "P $200":POSITION
 32,15:? "S $450":FOR I=1 TO N:M(I)=M(
I)-10:S$(I,I)=" ":NEXT I
544 POSITION 10,23:? "Place your bets:
";:X=12
550 FOR J=IP TO N:POSITION 1,X:? N$(J*
10-9,J*10);" - ";:X=X+1
560 IF J=N THEN B$=CHR$(65+RND(0)*11):
IF B$="K" THEN B$="P"
562 IF J=N THEN 580
570 GET #2,B:B$=CHR$(B):IF (B$>"@" AND
 B$<"K") OR B$="P" OR B$="S" THEN 580
572 GOTO 570
580 FOR K=1 TO N:IF B$=S$(K,K) THEN 56
0
582 NEXT K:PRINT B$;:S$(J,J)=B$:NEXT J
590 IF IP=1 THEN HN=1:GOTO 620
592 FOR J=1 TO IP-1:POSITION 1,X:? N$(
J*10-9,J*10);" - ";:X=X+1
600 GET #2,B:B$=CHR$(B):IF (B$>"@" AND
 B$<"K") OR B$="P" OR B$="S" THEN 610
602 GOTO 600
610 FOR K=1 TO N:IF B$=S$(K,K) THEN 60
0
612 NEXT K:PRINT B$;:S$(J,J)=B$:NEXT J
:HN=1
620 GOSUB 1470:POSITION 1,5:? O$(1,37)
:? O$(1,37):? O$(1,37):FOR X=1 TO 6:FO
R L=1 TO 5:D(X)=INT(RND(0)*6+1)
622 POSITION X*6-2,5:? D$(D(X)*17-16,D
(X)*17);:NEXT L:IF D(X)>HN THEN HN=D(X
)
624 NEXT X
628 FOR I=12 TO 15:POSITION 30,I:? "
        ";:NEXT I
630 FOR J=1 TO N:X=12+(J-IP):IF X<12 T
HEN X=X+N
631 POSITION 30,X:? "         ";:POSITI
ON 14,X:IF S$(J,J)<>"P" THEN 640
632 FOR K=1 TO 5:IF D(K)=D(K+1) THEN ?
 "$200 for a pair":M(J)=M(J)+200:K=5:G
OTO 690
634 NEXT K:? "Sorry, no pairs!":GOTO 6
90
```

```
640 IF S$(J,J)<>"S" THEN 660
642 FOR K=1 TO 4:IF D(K)=D(K+1)-1 AND
D(K)=D(K+2)-2 THEN ? "$450 for the str
aight":M(J)=M(J)+450:GOTO 690
644 NEXT K
650 ? "No straight!":GOTO 690
660 IF (S$(J,J)="G" AND (D(1)=HN OR D(
3)=HN)) OR (S$(J,J)="H" AND (D(3)=HN O
R D(5)=HN)) THEN 668
662 IF (S$(J,J)="I" AND (D(2)=HN OR D(
4)=HN)) OR (S$(J,J)="J" AND (D(4)=HN O
R D(6)=HN)) THEN 668
664 GOTO 670
668 ? "$150 for bet on ";S$(J,J):M(J)=
M(J)+150:GOTO 690
670 IF S$(J,J)<="@" OR S$(J,J)>="G" TH
EN 680
672 K=ASC(S$(J,J))-64:IF D(K)=HN THEN
PRINT "$300 on spot ";S$(J,J):M(J)=M(J
)+300:GOTO 690
674 PRINT "You lose with spot ";S$(J,J
);:GOTO 690
680 ? "Maybe next time!"
690 NEXT J
700 GOSUB 1470:RETURN
```

Lottery subroutine. Instructions and lottery tickets shown are in line 710. Array D(J) is set to zero. Six dice are rolled and displayed. If a number J is rolledED(J) is set to 1. Winning tickets are evaluated in line 730. Lottery ticket count L(I) is set to zero.

```
710 GRAPHICS 0:? "            L O T T E
 R Y":? :? "If both the numbers on you
r ticket are"
712 ? "among the six numbers rolled, y
ou will receive $50 for that ticket.":
? "Here we go..."
714 FOR J=1 TO N:POSITION 1,10+J:? N$(
J*10-9,J*10);"     ";L$(J*9-8,J*9-6);"
   ";L$(J*9-5,J*9-3);"   ";L$(J*9-2,J*
9)
716 NEXT J:GOSUB 1470
720 FOR J=1 TO 6:D(J)=0:NEXT J:FOR J=1
 TO 6:K=INT(RND(0)*6)+1:POSITION J*6-2
,7:? D$(K*17-16,K*17):D(K)=1
722 NEXT J
730 FOR J=1 TO N:IF L(J)=0 THEN 738
731 FOR K=1 TO L(J):FOR L=1 TO 150:NEX
T L
732 IF D(VAL(L$(J*9+K*3-11,J*9+K*3-11)
))=1 AND D(VAL(L$(J*9+K*3-9,J*9+K*3-9)
))=1 THEN 735
734 POSITION 10+K*6,10+J:? "---";:SOUN
D 0,220,10,10:FOR TI=1 TO 100:NEXT TI:
GOTO 737
735 POSITION 10+K*6,10+J:? "$50";:M(J)
=M(J)+50:FOR S=30 TO 10 STEP -2:SOUND
0,S,10,10:FOR TI=1 TO 2:NEXT TI
736 SOUND 0,255-S,10,10:FOR TI=1 TO 2:
NEXT TI:NEXT S
737 SOUND 0,0,0,0:NEXT K:L(J)=0
738 NEXT J:FOR J=1 TO N*9:L$(J,J)=" ":
NEXT J
740 GOSUB 1470:FOR J=1 TO N*9:L$(J,J)=
" ":NEXT J:L(1)=0:L(2)=0:L(3)=0:RETURN
```

Horse roace routine. The picture of a horse is in H$ in line 750. Heading and instructions are in lines 750-760. Horizontal position is in array D(K). Each of six horses is randomly advanced by incrementing its print position. When D(K) approaches the end of the screen in line 780, the winner is announced in lines 790 and 800.

```
750 GRAPHICS 0:POKE 752,1:H$=" _   u=+
++++__u_":FOR J=1 TO 6:D(J)=10:POSITIO
N 2,J*3+1:? J;"          ";H$;:NEXT J
751 REM H$ is typed as:[sp][shift-][sp
][sp][ctrlU][esc][ctrl=](5 times:[esc]
[ctrl+])[sp][sp][ctrlU][sp]
752 FOR J=1 TO N:POSITION 1,J*3+1:PRIN
T N$(J*10-9,J*10);:M(J)=M(J)-20:NEXT J
:POSITION 6,0:? "H O R S E   R A C E S
 !"
754 FOR Y=1 TO 6:POSITION 38,Y*3+1:? "
.";:NEXT Y
760 POSITION 1,21:? "Everyone has bet
$20, the winner will  receive $100.":G
OSUB 1470
770 SOUND 0,250,8,10:SOUND 0,240,8,10:
SOUND 0,255,8,10:SOUND 0,0,0,0
```

```
780 K=INT(RND(0)*6)+1:D(K)=D(K)+1:POSI
TION D(K),K*3+1:? H$;:IF D(K)<33 THEN
770
790 POSITION 1,21:? O$(1,38);:? O$(1,3
8);:? O$(1,38);:POSITION 1,21:IF K>N T
HEN 794
792 ? N$(K*10-9,K*10);" ";:M(K)=M(K)+1
00:GOTO 800
794 SOUND 0,0,0,0:? "Horse number ";K;
" ";
800 ? "wins this race!":GOSUB 1470:POK
E 752,0:RETURN
```

Fortune teller routine. The fortune teller's messsage is chosen at random in lines 810-880, and is displayed in lines 900-920.

```
810 M$="":GRAPHICS 0:POSITION 7,4:? "T
HE FORTUNE TELLER SAYS:":K=1:M=10*INT(
RND(0)*5+1)+50
820 IF RND(0)<0.2 THEN M$="Hold a Swee
pstakes.":K=2:GOTO 890
830 IF RND(0)<0.2 THEN M$="Hold a Lott
ery.":K=3:GOTO 890
840 IF RND(0)<0.5 THEN M$="Collect fro
m ":K=-1:GOTO 850
842 M$="Pay to "
850 J=INT(RND(0)*(N+1)+1):IF J<>IP THE
N 860
852 M$(LEN(M$)+1)="everyone":M(IP)=M(I
P)-K*N*M:FOR J=1 TO N:M(J)=M(J)+K*M:NE
XT J:GOTO 880
860 IF J=N+1 THEN M$(LEN(M$)+1)="the b
ank":M(IP)=M(IP)-K*M:GOTO 880
870 M$(LEN(M$)+1)=N$(J*10-9,J*10):M(IP
)=M(IP)-K*M:M(J)=M(J)+K*M
880 M$(LEN(M$)+1)=" ":M$(LEN(M$)+1)=ST
R$(M):M$(LEN(M$)+1)=" dollars."
890 QQ=0
900 FOR TI=LEN(M$) TO 1 STEP -1:M$(TI+
44,TI+44)=M$(TI,TI):NEXT TI:M$(1,32)=O
$(1,32):M$(33,42)=N$(IP*10-9,IP*10)
901 M$(43,44)=", ":M$(LEN(M$)+1)=O$(1,
32)
902 FOR J=1 TO LEN(M$)-31:SOUND 0,200,
8,10:POSITION 5,6:? M$(J,J+30);:SOUND
0,0,0,0:FOR TI=1 TO 5:NEXT TI:NEXT J:M
$=""
910 GOSUB 1470
930 IF K=2 THEN 460
932 IF K=3 THEN 710
934 RETURN
```

High roller game. Instructions are in line 940; all roll two dice in line 950. High roll is determined in lines 960 and 970. Single winner is announced in line 980; a tie causes a re-roll in lines 990-1010.

```
940 GRAPHICS 0:? "H I G H   R O L L E
R -- Everyone      antes $20 and rolls
 two dice.  The"
942 ? "highest roll takes the $100 pri
ze.  To roll dice, touch any key on yo
ur turn.":GOSUB 1470
950 FOR J=1 TO N:M(J)=M(J)-20:POSITION
 1,2+J*3:? N$(J*10-9,J*10):X1=12+(J-IN
T(J/2)*2)*4:X=2+J*3:GOSUB 1450:D(J)=K
952 X1=X1+8:GOSUB 1450:D(J)=D(J)+K:POS
ITION 28,X:? "Total is ";D(J);:NEXT J
960 HN=D(1):HM=1:FOR J=2 TO N:IF D(J)>
HN THEN HN=D(J):HM=J
962 NEXT J
970 K=0:FOR J=1 TO N:IF D(J)=HN THEN K
=K+1
972 NEXT J
980 IF K=1 THEN POSITION 1,20:? N$(HM*
10-9,HM*10);" wins the pot!":M(HM)=M(H
M)+100:POKE 764,255:GOSUB 1470:RETURN
990 POSITION 1,20:? "We've got a tie!
 Those high rollers   will roll again!
":POKE 764,255:GOSUB 1470
992 POSITION 1,5:FOR J=1 TO 18:? O$;:N
EXT J
1000 FOR J=1 TO N:X1=14:X=2+3*J:IF D(J
)<HN THEN D(J)=0:NEXT J:GOTO 960
1002 POSITION 1,X:? N$(J*10-9,J*10):X1
=12+(J-INT(J/2)*2)*4:GOSUB 1450:D(J)=K
:X1=X1+8:GOSUB 1450
1004 D(J)=D(J)+K:POSITION 28,X:? "Tota
l is ";D(J);:X=X+3:NEXT J
1010 GOTO 960
```

Poker party subroutine. Instructions are in line 1020. Three dice are rolled in line 1030 for all players. Check for three of a kind in line 1040; for pairs in 1050; for straights in 1070. Winner is announced in 1120. A tie causes the routine to restart.

```
1020 GRAPHICS 0:? "P O K E R  P A R T
 Y -- Each player  pays $20 and rolls
 three dice.  The"
1022 ? "best poker hand":? "(Three of
a Kind > Straight > Pair)   wins $100
.  Touch any key to roll dice.";
1024 GOSUB 1470:XX=6:FOR I=1 TO N:M(I)
=M(I)-20:NEXT I:HM=0
1030 FOR I=1 TO N:POSITION 1,XX:? N$(I
*10-9,I*10):FOR M=1 TO 3:J=I:X1=9+M*6:
X=XX:POKE 764,255:GOSUB 1450:D(M)=K
1032 NEXT M:P(I)=0
1040 FOR TI=1 TO 100:NEXT TI:FOR TI=X
TO XX+2:POSITION 11,TI:PRINT O$(1,28);
:NEXT TI:POSITION 13,XX
1041 ? D(1);" ";D(2);" ";D(3);"  ";
1042 IF D(1)=D(2) AND D(1)=D(3) THEN ?
 "Three of a kind!";:P(I)=30+D(1):GOTO
 1100
1050 IF D(1)=D(2) THEN P(I)=10+D(1)
1052 IF D(1)=D(3) THEN P(I)=10+D(1)
1054 IF D(2)=D(3) THEN P(I)=10+D(2)
1060 IF P(I)>0 THEN PRINT "A pair!";:G
OTO 1100
1070 FOR J=1 TO 3:IF (D(1)=D(2)+1 AND
D(1)=D(3)+2) OR (D(1)=D(2)-1 AND D(1)=
D(3)-2) THEN P(I)=20:GOTO 1080
1072 HN=D(1):D(1)=D(2):D(2)=D(3):D(3)=
HN:NEXT J
1080 HN=D(1):FOR J=2 TO 3:IF D(J)>HN T
HEN HN=D(J)
1082 NEXT J
1090 P(I)=P(I)+HN:IF P(I)>20 THEN PRIN
T "A straight!";:GOTO 1100
1092 PRINT "Highest roll is a ";P(I);
1100 XX=XX+2:IF P(I)>HM THEN HM=P(I)
1102 NEXT I
1110 K=0:FOR I=1 TO N:IF P(I)=HM THEN
K=K+1:J=I
1112 NEXT I
1120 IF K=1 THEN POSITION 1,20:? N$(J*
10-9,J*10);" wins the pot of $100":M(J
)=M(J)+100:GOSUB 1470:RETURN
1122 POSITION 1,20:? "We have a tie ..
. so let's all play   another hand!":
GOSUB 1470:GOTO 1020
```

Love thy neighbor routine. Set the message in line 1150; adjust money in line 1160; jump to display routine in line 890.

```
1130 GRAPHICS 0:K=0:M=INT(RND(0)*5+1)*
10+50
1140 J=INT(RND(0)*N+1):IF J=IP THEN 11
40
1150 M$="Show that you are a good neig
hbor and give ":M$(LEN(M$)+1)=N$(J*10-
9,J*10):M$(LEN(M$)+1)=" "
1152 M$(LEN(M$)+1)=STR$(M):M$(LEN(M$)+
1)=" dollars."
1160 M(IP)=M(IP)-M:M(J)=M(J)+M:GOTO 89
0
```

Easy come, easy go. Give instructions and roll two dice in line 1170. Adjust money.

```
1170 GRAPHICS 0:? "E A S Y  C O M E
 E A S Y  G O  -- The bank will pay
you 10 times the roll";
1172 ? "of 2 dice.  Touch any key to r
oll dice,";N$(IP*10-9,IP*10):GOSUB 147
0:J=IP:X1=15:X=6:GOSUB 1450:M=K
1174 X1=X1+6:GOSUB 1450:M=(M+K)*10:M(I
P)=M(IP)+M
1176 POSITION 7,10:? "You win ";M;" do
llars."
1180 GOSUB 1470:RETURN
```

Short messages routines. Simple procedures to handle special routines: Win A Few (1190), Unlucky Seven (1200), Lose A Few (1230), Even Steven (1240), Pot Luck (1300), Jackpot (1310), $100 Bonus (1320), Lose A Turn (1330), Tax Time (1340), Bonanza (1350), and Mad Money (1360).

```
1190 GRAPHICS 0:? "W I N  A  F E W
```

```
-- The bank will payyou ten times the
roll of one die."
1192 ? "Touch any key to roll die, ";N
$(IP*10-9,IP*10):GOSUB 1470:J=IP:X1=18
:X=6:GOSUB 1450:M=K*10:M(IP)=M(IP)+M
1194 GOTO 1176
1200 GRAPHICS 0:? "U N L U C K Y   S E
 V E N  -- Roll two dice.  If the tota
l is seven you lose"
1202 ? "$100.  For any other total, yo
u win    $100.  Touch any key to roll
dice,":? N$(IP*10-9,IP*10):GOSUB 1470
1204 J=IP:X1=15:X=6:GOSUB 1450:M=K:X1=
X1+6:GOSUB 1450:M=M+K
1210 POSITION 7,10:IF M=7 THEN ? "You
lose 100 dollars!":M(IP)=M(IP)-100:GOT
O 1220
1212 ? "You win 100 dollars!":M(IP)=M(
IP)+100
1220 GOSUB 1470:RETURN
1230 GRAPHICS 0:? "L O S E   A   F E W
  -- You must pay   the bank ten times
 the roll of one die.";
1232 ? "Touch any key to roll die,":?
N$(IP*10-9,IP*10):GOSUB 1470:J=IP:X1=1
8:X=6:GOSUB 1450:M=K*10
1234 M(IP)=M(IP)-M:POSITION 7,10:? "Yo
u lose ";M;" dollars.":GOSUB 1470:RETU
RN
1240 GRAPHICS 0:? "E V E N   S T E V E
 N  -- You may bet  up to $90 and roll
 two dice.  If the"
1242 ? "total is even, you collect twi
ce your  bet.  Touch any key to roll d
ice."
1244 ? N$(IP*10-9,IP*10):? "How much w
ill you bet ? ";:J=IP:IF IP=N THEN B=5
4+INT(RND(0)*4):GOTO 1260
1250 GET #2,B:IF B<49 OR B>57 THEN 125
0
1260 M=(B-48)*10:? M;" dollars":GOSUB
1470
1270 X1=15:X=7:GOSUB 1450:P=K:X1=X1+6:
GOSUB 1450:P=P+K
1280 POSITION 7,11:IF INT(P/2)*2=P THE
N ? "You win ";2*M;" dollars.":M(IP)=M
(IP)+2*M:GOTO 1290
1282 ? "You lose ";M;" dollars.":M(IP)
=M(IP)-M
1290 GOSUB 1470:RETURN
1300 GRAPHICS 0:M=10*INT(RND(0)*5+1)+1
0:M(IP)=M(IP)-M:M$="Why don't you swee
ten the pot by giving the bank "
1302 M$(LEN(M$)+1)=STR$(M):M$(LEN(M$)+
1)=" dollars.":K=0:GOTO 890
1310 GRAPHICS 0:M=10*INT(RND(0)*5+1)+5
0:M$="You hit the Jackpot!  Collect ":
M$(LEN(M$)+1)=STR$(M)
1312 M$(LEN(M$)+1)=" dollars.":K=0:M(I
P)=M(IP)+M:GOTO 890
1320 GRAPHICS 0:M(IP)=M(IP)+100:M$="Pl
ease accept this $100 bonus!":K=0:GOTO
 890
1330 GRAPHICS 0:M$="You just lost this
 turn!":K=0:GOTO 890
1340 GRAPHICS 0:M$="Pay $100 tax to th
e bank!":M(IP)=M(IP)-100:K=0:GOTO 890
1350 GRAPHICS 0:M=INT(RND(0)*5+1)*10:M
(IP)=M(IP)+M*N:FOR J=1 TO N:M(J)=M(J)-
M:NEXT J
1352 M$="What a Bonanza!  Everyone pay
s you ":M$(LEN(M$)+1)=STR$(M):M$(LEN(M
$)+1)=" dollars.":K=0:GOTO 890
1360 GRAPHICS 0:M=INT(RND(0)*5+1)*10:M
(IP)=M(IP)-M*N:FOR J=1 TO N:M(J)=M(J)+
M:NEXT J
1362 M$="This ought to make you mad.
Give everybody ":M$(LEN(M$)+1)=STR$(M)
:M$(LEN(M$)+1)=" dollars.":K=0:GOTO 89
0
```

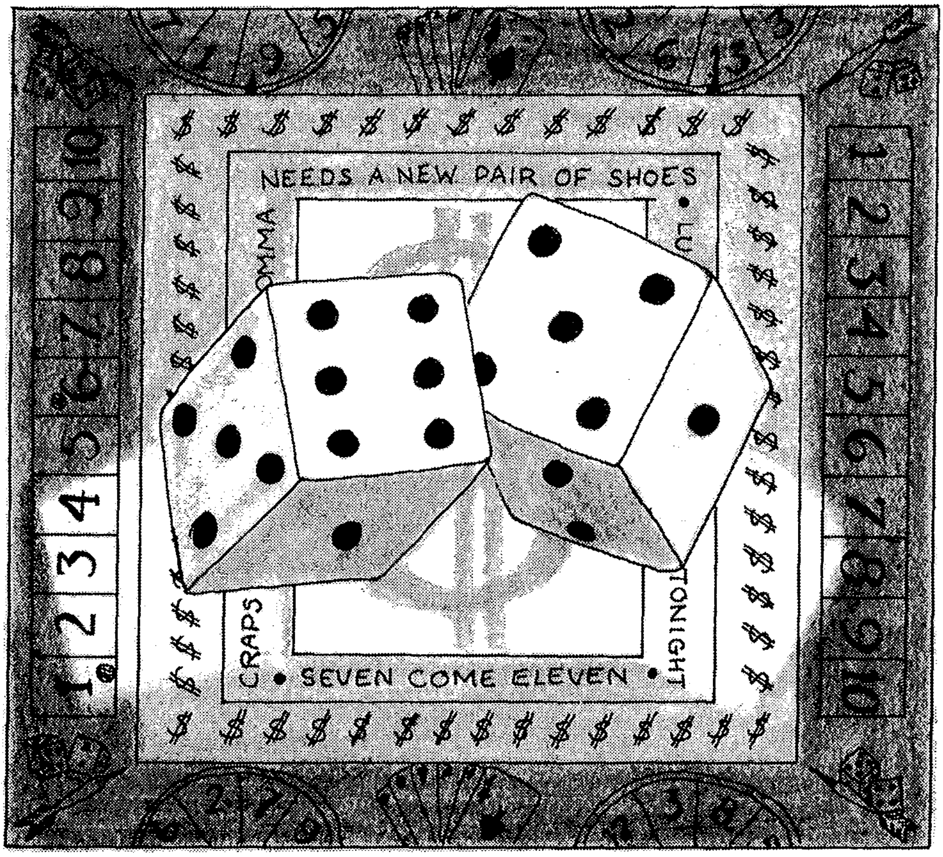
Instructions.

```
1370 GRAPHICS 0:? "This is the game of
 GAMBLER!":? :? "You and the Atari are
 in a contest to"
1372 ? "see who will build his $100 ba
nkroll   into $1000 first.  Money is m
ade and"
1374 ? "lost through a series of games
```

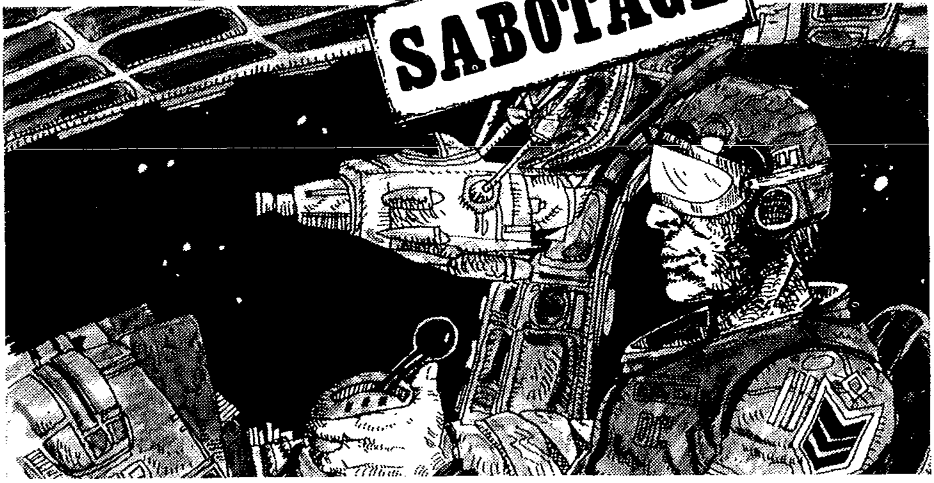
```
 of      chance -- from horse racing a
nd dice"
1380 ? "games to lotteries and sweepst
akes!    If you should lose all your m
oney, your"
1382 ? "IOU will be accepted (as long
as you   pay it back with interest).":
? :? "              GOOD LUCK!":K=0:GOTO
910
```

Utility subroutines. Lines 1540-1460 wait for input before stopping on one of the six dice at screen position X1, X. The computer's dice choice is made in line 1450. Lines 1470 and 1480 print a message a wait for keyboard response.

```
1450 POKE 764,255:S=PEEK(752):POKE 752
,1
1452 IF J<>N THEN 1460
1454 FOR L=1 TO RND(0)*50+1:K=INT(RND(
0)*6+1):POSITION X1,X:? D$(K*17-16,K*1
7);:POKE 53279,RND(0)*4:NEXT L:GOTO 14
62
1460 B=PEEK(764):K=INT(RND(0)*6+1):POS
ITION X1,X:? D$(K*17-16,K*17);:POKE 53
279,RND(0)*4:IF B=255 THEN 1460
1462 POKE 764,255:POKE 752,S:RETURN
1469 STOP
1470 POKE 764,255:POSITION 1,23:? "
  Touch any key to continue";
1475 IF RND(0)<0.11 THEN S=RND(0)*70+3
0:FOR TI=S TO S-8 STEP -4:SOUND 0,TI,1
0,10:NEXT TI:SOUND 0,0,0,0
1480 IF PEEK(764)=255 THEN 1475
1482 POKE 764,255:POSITION 6,23:? "
                          ";:RETURN
```



Operation SABOTAGE



by Ray Sato
Atari version by Rich Bouchard

Encryption modifications by Rich Bouchard, William Kubeck, and Alan J. Zett

***Operation: Sabotage* is an adventure game for an Atari 400/800 with 24K RAM (tape) or 32K (disk).**

It is the year 2101 and war has broken out between Earth and the distant planet Zekloke. This alien power has established a large military complex on Mars which will soon become a great danger to Earth. Hidden in the massive installation are several secret documents containing the plans for an incredible defense shield — strong enough to stop an entire fleet of spacecraft.

You are a special agent and have just succeeded in sneaking into the alien complex. Your mission is to destroy this threat to mankind and return with plans for the powerful defense shield. The outcome of this mission will decide the fate of mankind.

Playing Notes

The computer will always give you a brief description of where you are, what objects you can see, and what exits are visible. You move and act by typing in simple commands, generally consisting of a verb and a noun. If the computer tells you that there is a laser pistol in the room, for example, you might want to type in the command "GET PISTOL". At a later time, you might be able to use it to "SHOOT MONSTER" or for some other purpose. If you no longer want to carry it, you can "DROP PISTOL" whenever you please. Since the computer looks only at the first three letters of the verb and the last three letters of the noun, you may use abbreviations such as "SHO TER" (for "SHOOT MONSTER") if you desire. Movement is accomplished by entering just a single letter rather than a two-word command: N, S, E, or W for north, south, east,or west. Typing the single word "INVENTORY" (or "INV") will display a list of what you are carrying. Typing "STATUS" (or "STA") will give you a readout of your current physical condition.

Part of the challenge of any adventure game such as *Operation: Sabotage* is to figure out what you are able to do in a particular situation. Therefore, you will not find a list of all the verbs the computer can understand, or of all the objects you may discover. You might find yourself frustrated by what seem to be dead-ends, and end up getting killed in the process. This is all part of the adventure, and a test of your ingenuity and perseverance.

Program Notes

The most obvious feature of the program listing is that most of it looks like a cryptogram. The BASIC keywords are all in their usual form, but the string assignment statements and DATA lines are incomprehensible. This is because all of the room descriptions, object names, monsters, and verbs have been encoded. This has been done to preserve the value of the game. Anyone who types an adventure program in from a listing is bound to be disappointed in the game's playability, since he has gained so many clues about the plot. So, even though the typing is made slightly difficult by the scrambled words, this is the only reasonable way of publishing adventure programs in listed form. We have also omitted the usual list of variables for the same reasons. The variable descriptions give away too much information and the encoding of the program reduces the usefulness of a variable list.

The encryption method is a simple one, which results in leaving punctuation unmodified, and inverting the order of the letters of the alphabet. This simple inversion process has the advantage of using the same routine to decode the text as was used to encode it. In *Operation: Sabotage*, the user's input is encoded, the internal searches and comparisons are done in encoded form, and the response is decoded and printed by the subroutine at line 5.

SWAT

In order to offset the proofreading problems created by this approach, we have included an expanded *SWAT* Table for this program. Instead of the normal 12-line/500-byte *SWAT* parameters, we have used 5-line/200-byte parameters. This means that you must modify the first line of the *SWAT* program in order to generate a table to compare with ours. After merging *SWAT* in the normal manner, but before running it, simply edit or retype line 32000, changing "NU=12: B=500" to "NU=5: B=200". This will provide an expanded *SWAT* Table, enabling you to pinpoint typing mistakes more easily.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC         SS
SS    'Operation Sabotage'     SS
SS      Author: Ray Sato       SS
SS Translator:  Rich Bouchard SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Jump to program initialization.

```
1 GOTO 2620
```

Decode and print output.

```
4 IF P$="" THEN RETURN
5 FOR P=N1 TO LEN(P$):? CHR$(ABS((155*
(P$(P)>"@"))-ASC(P$(P))));:NEXT P:? :R
ETURN
```

Encode input.

```
6 V$="":IF VO$="" THEN RETURN
7 FOR J=N1 TO LEN(VO$):V$(LEN(V$)+N1)=
CHR$(ABS((155*(VO$(J)>"@"))-ASC(VO$(J)
))):NEXT J:RETURN
```

Auxiliary jump points.

```
8 GOSUB N4:GOTO 2520
9 GOSUB N4:GOTO L1
```

Descriptions of individual rooms.

```
10 A$="ZM ZRIOLXP.  GSVIV RH Z    YOFV
 YFGGLM SVIV":S=N2:RETURN
20 A$="Z MZIILD XLIIRWLI":N=N1:S=N3:RE
TURN
30 A$="Z MZIILD XLIIRWLI":N=N2:S=N4:RE
TURN
40 A$="Z MZIILD XLIIRWLI":N=N3:S=N5:RE
TURN
```

```
50 A$="Z HNZOO ILLN":N=N4:S=6:RETURN
60 A$="Z WVXLMGZNRMZGRLM XSZNYVI":B$="
GSVIV RH Z YOFV YFGGLM SVIV":N=N5:S=7:
RETURN
70 A$="Z HNZOO HGLIZTV XSZNYVI":N=6:S=
N8:W=12:RETURN
80 A$="Z HNZOO XSZNYVI":N=7:S=N9:W=13:
RETURN
90 A$="Z HNZOO VOVXGILMRX          OZYL
IZGLIB":N=N8:S=10:W=14:RETURN
100 A$="Z YRLOLTRXZO OZYLIZGLIB.   GSV
IV RH Z IVW YFGGLM LM GSV DZOO":N=N9:W
=15:RETURN
110 A$="Z HGLIZTV XSZNYVI":W=16:RETURN
120 A$="Z OZITV XSZNYVI.  GSVIV RH Z X
ZYRMVG SVIV":S=13:W=17:E=7:RETURN
130 A$="Z HGIZMTV KFIKOV ILLN.     GSV
IV RH Z YOFV YFGGLM SVIV":N=12:S=14:W=
18:E=N8:RETURN
140 A$="Z HNZOO LUURXV":N=13:S=15:E=N9
:RETURN
150 A$="Z HNZOO ILLN DRGS Z XZIW   GZY
OV RM GSV XVMGVI":N=14:S=16:W=20:E=10:
RETURN
160 A$="Z OZITV LUURXV.  GSVIV RH ZWVH
P SVIV":N=15:W=21:E=11:RETURN
170 A$="Z LUURXV DRGS Z OZITV WVHP":S=
18:E=12:RETURN
180 A$="Z HGLIZTV ILLN":N=17:S=19:E=13
:RETURN
190 A$="Z OZITV SZOO":N=18:S=20:RETURN
200 A$="ZM VMGVIGZRMNVMG ILLN.  Z  HXI
VVM IVHGH LM GSV DZOO":B$="GSVIV RH Z
YOFV ZMW Z IVW YFGGLM MVZI GSV HXIVVM"
202 N=19:S=21:E=15:RETURN
210 A$="Z WZGZ IVXLIW HGLIZTV ILLN":N=
20:E=16:RETURN
220 A$="IZWZI XLMGILO.  GSVIV RH Z HNZ
OO HXIVVM SVIV":S=23:W=27:RETURN
230 A$="Z NVWRXZO HGZGRLM.  GSVIV  RH
Z OZITV GZYOV SVIV":N=22:S=24:W=28:RET
URN
240 A$="Z HVXFIRGB HGZGRLM":N=23:RETUR
N
250 A$="Z IZWRL ILLN":S=26:W=30:RETURN
260 A$="Z HNZOO ILLN.  GSVIV RH Z  HZU
V RM GSV HLFGS DZOO":N=25:W=31:RETURN
270 A$="GSV ILYLG XLMGILO XVMGVI.  GSV
IV RH Z HNZOO XLMGILO XLNKFGVI     NLF
MGVW RM GSV DZOO":B$="":W=32:E=22
272 RETURN
280 A$="GSV DVZKLMH HGLIZTV ILLN":S=29
:W=33:E=23:RETURN
290 A$="Z ORYIZIB":N=28:S=30:W=34:RETU
RN
300 A$="Z HVXFIRGB XSVXP ZIVZ":N=29:W=
36:E=25:RETURN
310 A$="Z HNZOO ILLN DRGS Z WVHP.  Z H
RTM IVZWH"
312 B$="'KIVHHFIV GL IVZXGLI.'  Z YOFV
 YFGGLM RH OLXZGVW FMWVI GSV HRTM":E=2
6:RETURN
320 A$="OZFMXS XLMGILO.  GSVIV RH ZWVH
P SVIV":S=33:W=37:E=27:RETURN
330 A$="Z HNZOO XLIIRWLI":N=32:S=34:E=
28:RETURN
340 A$="Z HGIZMTV YOFV ILLN.  GSVIVRH
Z IVW YFGGLM SVIV":N=33:S=35:E=29:RETU
RN
350 A$="Z GRMB HGLIZTV ILLN":N=34:W=40
:RETURN
360 A$="Z HNZOO XSZNYVI.  Z WLLI   DVH
G OVZWH GL GSV MFXOVZI IVZXGLI":E=30:R
ETURN
370 A$="Z HNZOO, MZIILD XLIIRWLI":N=42
:E=32:RETURN
380 A$="GSV XLNKFGVI XVMGVI.  GSVIVRH
Z HNZOO HOLG RM GSV XLNKFGVI":S=39:RET
URN
390 A$="GSV XSVNRXZO OZY.":N=38:S=40:R
ETURN
400 A$="GSV IVZXGLI XLMGILO XVMGVI.GSV
IV RH Z YOFV YFGGLM ZMW Z IVW LMV. Z H
RTM HZBH IVZXGLI XLMGILO - "
402 A$(LEN(A$)+N1)="IVW=LM, YOFV=LUU":
N=39:E=35:RETURN
410 A$="GSV MFXOVZI IVZXGLI.  Z    XLN
KFGVI IVHGH LM GSV DZOO":E=36:RETURN
420 A$="GSV DVHG VMW LU Z OLMT     XLI
IRWLI":S=37:E=43:RETURN
```

```
430 A$="GSV VZHG VMW LU Z OLMT     XLI
IRWLI":W=42:E=44:RETURN
440 A$="Z HVXFIRGB XVMGVI":W=43:E=45:R
ETURN
450 A$="Z HNZOO OZFMXS ZIVZ":B$="GSVIV
 RH Z HNZOO HOLG MVCG GL GSV     OZFMX
STZGV":W=44:RETURN
```

Extended descriptions.

```
460 REM
470 IF A=10 AND (D3=N1 OR D3=N2) THEN
C$="GSV NLMHGVI'H XZTV RH LKVM"
480 IF A=12 AND D5=NO THEN C$="GSV XZY
RMVG RH OLXPVW"
490 IF A=12 AND D5=N1 THEN C$="GSV XZY
RMVG RH LKVM"
500 IF A=20 AND D6=NO THEN C$="GSV HXI
VVM RH YOZMP"
510 IF A=20 AND D6=N1 THEN C$="Z NLERV
 RH YVRMT KOZBVW LM GSV HXIVVM"
520 IF A=26 AND D9=NO THEN C$="GSV HZU
V RH OLXPVW"
530 IF A=26 AND D9=N1 THEN C$="GSV HZU
V RH LKVM"
540 IF A=27 AND E2=NO THEN C$="GSV XLN
KFGVI RH ZXGREV"
550 IF A=27 AND E2=N1 THEN C$="GSV XLN
KGVI RH WVHGILBVW"
560 IF A=36 AND E6=NO THEN C$="GSV IVZ
XGLI WLLI RH URINOB OLXPVW"
570 IF A=36 AND E6=N1 THEN C$="GSV IVZ
XGLI WLLI RH LKVM":W=41
580 IF A=45 AND E9=NO THEN C$="GSV OZF
MXS TZGV RH XOLHVW"
590 IF A=45 AND E9=N1 THEN C$="GSV OZF
MXS TZGV RH LKVM":E=46
```

Generate the list of visible items and available exits.

```
600 A$(LEN(A$)+N1)=".":IF LEN(B$)>N3 T
HEN B$(LEN(B$)+N1)="."
610 IF LEN(C$)>N3 THEN C$(LEN(C$)+N1)=
"."
650 IF N<>NO THEN E$="MLIGS "
660 IF S<>NO THEN E$(LEN(E$)+N1)="HLFG
S "
670 IF W>NO THEN E$(LEN(E$)+N1)="DVHG
"
680 IF E>NO THEN E$(LEN(E$)+N1)="VZHG
"
690 IF E$<>"" THEN E$=E$(N1,LEN(E$)-N1
)
```

Describe current location, visible items, and available exits.

```
700 GRAPHICS NO:PRINT "YOU ARE IN ";:P
$=A$:GOSUB N4:PRINT :IF B$<>"" THEN P$
=B$:GOSUB N4
710 IF C$<>"" THEN P$=C$:GOSUB N4
720 P$=" ":PRINT :PRINT "OBJECTS YOU C
AN SEE:":FOR T=N1 TO 16
722 IF A=I(T) THEN P$=I$(IP(T),IP(T+N1
)-N1):GOSUB N4
730 NEXT T:IF P$=" " THEN P$="MLGSRMT"
:GOSUB N4
732 ? :? "EXITS: ":P$=E$:GOSUB N4:?
740 IF (A=40 OR A=35 OR A=30 OR A=31)
AND I(14)=NO AND F3=NO THEN P$="GSV HN
ZOO YOZXP WVERXV RH YORMPRMT":GOSUB N4
750 IF A=36 AND I(N4)=NO AND F3=NO THE
N P$="GSV HNZOO YOZXP WVERXV RH UOZHSR
MT    YIRTSGOB":GOSUB N4
760 IF F4=NO THEN 770
762 P$="GSV XLNKFGVI HZBH:":GOSUB N4:P
$=STR$(F4):P$(LEN(P$)+N1)=" NRMFGVH FM
GRO WVHGIFXGRLM":GOSUB N4
770 IF D3=N1 THEN PRINT :P$="* * * ZOR
VM NLMHGVI ZGGZXPRMT * * *":GOSUB N4
780 IF D7=N1 OR E0=N1 OR E3=N1 OR E7=N
1 THEN PRINT :P$="* * * HVXFIRGB KZGIL
O ZGGZXPRMT * * *":GOSUB N4
```

Get and interpret command.

```
790 PRINT :PRINT "COMMAND";:INPUT V0$:
GOSUB 6:PRINT
800 FOR T=N1 TO N4:IF V$=VB$(T*N3-N2,T
*N3-N2) THEN V$=VB$(T*N3-N2,T*N3)
810 NEXT T
820 IF LEN(V$)<N3 THEN 700
830 V1$=V$(N1,N3):V2$=V$(LEN(V$)-N2)
840 FOR T=N1 TO 17:IF V1$=VB$(T*N3-N2,
T*N3) THEN V1=T
```

```
850 NEXT T:IF V1=N0 THEN P$="R WLM'G F
MWVIHGZMW DSZG BLF DZMG.":GOSUB N4:GOT
O L1
860 FOR T=N1 TO 16:IF V2$=I$(IP(T+N1)-
N3,IP(T+N1)-N1) THEN V2=T
870 NEXT T
880 ON V1 GOTO 900,940,980,1050,1130,1
310,1370,1390,1560,1620,1660,1790,1860
,1910,2010,2170,2250
890 GOTO L1
```

Handle commands.

```
900 IF N=N0 THEN GOTO L2
910 IF D3=N1 THEN P$="GSV NLMHGVI YOLX
PH GSV VCRG":GOSUB N4:GOTO L2
920 IF D7=N1 OR E3=N1 THEN GOTO L2
930 A=N:GOTO L1
940 IF S=N0 THEN GOTO L2
950 IF S=24 AND D8<>N0 AND E2<>N1 THEN
 D7=N1:GOSUB L4
960 IF S=30 AND E4<>N0 AND E2<>N1 THEN
 E3=N1:GOSUB L4
970 A=S:GOTO L1
980 IF W=N0 THEN GOTO L2
990 IF D3=N1 THEN P$="GSV NLMHGVI YOLX
PH GSV VCRG":GOSUB N4:GOTO L1
1000 IF E0=N1 OR E3=N1 OR E7=N1 THEN G
OTO L3
1010 IF W=41 AND F3=N0 THEN P$="IZWRZG
RLM UILN GSV IVZXGLI SRGH BLF":GOSUB N
4:GOTO 2520
1020 IF W=30 AND E4<>N0 AND E2<>N1 THE
N E3=N1:GOSUB L4
1030 IF W=27 AND E8<>N0 AND E2<>N1 THE
N E0=N1:GOSUB L4
1040 A=W:GOTO L1
1050 IF E=N0 THEN GOTO L2
1060 IF E0=N1 OR E3=N1 OR E7=N1 THEN G
OTO L3
1070 IF E=27 AND E1<>N0 AND E2<>N1 THE
N E0=N1:GOSUB L4
1080 IF E=44 AND E8<>N0 AND E2<>N1 THE
N E7=N1:GOSUB L4
1090 A=E:GOTO L1
1100 P$="GSV HVXFIRGB TFZIW YOLXPH GSV
 VCRG":GOSUB N4:GOTO L1
1110 P$="GSVIV RH ML DZB GL TL GSZG WR
IVXGRLM":GOTO N9
1120 P$="Z HVXFIRGB ZMWILRW ZDZRGH BLF
":GOSUB N4:RETURN
1130 IF A<>N1 OR V2$<>"LXP" THEN 1140
1132 P$="GSV ZRIOLXP LKVMH ZMW BLF ZIV
 YOLDM   LFG RMGL GSV EZXFFN LU HKZXV"
:GOSUB N4:GOTO 2520
1140 IF A<>12 OR V2$<>"MVG" OR D5<>N0
OR I(N2)=N0 THEN 1150
1142 P$="BLF QFHG ZIVM'G HGILMT VMLFTS
 GL ULIXVRG LKVM":GOSUB N4:GOTO L1
1150 IF A<>12 OR V2$<>"MVG" OR D5<>N0
OR I(N2)<>N0 THEN 1160
1152 P$="GSV XILDYZI SVOKVW.  GSV XZYR
MVG RH   MLD LKVM":GOSUB N4:D5=N1:I(N5
)=ABS(I(N5)):GOTO L1
1160 IF A=12 AND V2$="MVG" AND D5=N1 T
HEN P$="GSV XZYRMVG RH ZOIVZWB LKVM":G
OSUB N4:GOTO L1
1170 IF A=16 AND V2$="VHP" THEN ? OK$:
I(6)=ABS(I(6)):GOTO L1
1180 IF A=17 AND V2$="VHP" THEN PRINT
OK$:I(7)=ABS(I(7)):GOTO L1
1190 IF A=26 AND V2$="ZUV" AND D9=N1 T
HEN P$="GSV HZUV RH ZOIVZWB LKVM":GOTO
 N4:GOTO L1
1200 IF A=26 AND V2$="ZUV" AND D9=N0 T
HEN P$="R WLM'G SZEV GSV PVB GL LKVM G
SV HZUV":GOTO N9
1210 IF A=31 AND V2$="VHP" THEN P$="LP
   BLF URMW MLGSRMT RMHRWV":GOTO N9
1220 IF A=32 AND V2$="VHP" THEN PRINT
OK$:I(14)=ABS(I(14)):GOTO L1
1230 IF A=36 AND V2$="LLI" AND E6=N1 T
HEN P$="GSV WLLI RH ZOIVZWB LKVM":GOTO
 N9
1240 IF A<>36 OR V2$<>"LLI" OR E6<>N0
THEN 1260
1242 IF I(6)<>N0 THEN P$="BLF WLM'G SZ
EV GSV PVB GL GSV WLLI":GOTO N9
1250 IF E5=N0 THEN P$="GSV IVZXGLI YFR
OWRMT RH MLG           KIVHHFIRAVW.  B
LF ZIV YOLDM RMGL GSV  EZXFFN":GOTO N8
1260 IF A=36 AND V2$="LLI" AND I(6)=N0
```

```
 THEN P$="GSV WLLI RH MLD LKVM":E6=N1:
GOTO N9
1270 IF A=41 AND V2$="MVO" THEN P$="GS
V KZMVO RH URINOB OLXPVW":GOTO N9
1280 IF A=45 AND V2$="LXP" AND E9=N1 T
HEN P$="GSV ZRIOLXP RH ZOIVZWB LKVM":G
OTO N9
1290 IF A=45 AND V2$="LXP" AND E9=N0 T
HEN P$="GSVIV ZIVM'G ZMB ERHRYOV XLMGI
LOH":GOTO N9
1300 P$="R XZM'G WL GSZG":GOTO N9
1310 IF V2$="GVI" OR V2$="LRW" THEN P$
="WLM'G YV IRWRXFOLFH":GOTO N9
1320 IF V2=N0 THEN P$="R XZM'G WL GSZG
":GOTO N9
1330 IF I(V2)=N0 THEN P$="BLF ZOIVZWB
SZEV GSZG":GOTO N9
1340 IF A<>I(V2) THEN P$="R WLM'G HVV
RG SVIV":GOTO N9
1350 IF P4>=N8 THEN P$="HLIIB, BLF XZM
'G XZIIB ZMBGSRMT NLIV":GOSUB N4:GOTO
L1
1360 P4=P4+N1:I(V2)=N0:PRINT OK$:GOTO
L1
1370 IF V2=N0 THEN P$="BLF WLM'G SZEV
GSZG":GOTO N9
1380 P4=P4-N1:I(V2)=A:PRINT OK$:GOTO L
1
1390 IF I(N5)<>N0 THEN P$="BLF WLM'G S
ZEV Z DVZKLM":GOTO N9
1400 IF A=N1 AND V2$="LXP" THEN P$="BL
F ZIV YOLDM LFG LU GSV ZRIOLXP RMGL GS
V EZXFFN LU HKZXV":GOTO N8
1410 IF A=27 AND V2$="GVI" THEN P$="GS
V XLNKFGVI RH WVHGILBVW":E2=N1:E0=N0:G
OTO N9
1420 IF A=38 AND V2$="GVI" THEN P$="GS
V HSLG IVUOVXGH LUU LU GSV XLNKFGVI":G
OTO N8
1430 IF A=41 AND V2$="GVI" THEN P$="GS
V DSLOV MFNXOVZI IVZXGLI RH VCKOLWRMT"
:GOTO N8
1440 IF V2$="RWH" OR V2$="YLG" OR V2$=
"ILO" OR V2$="IWH" OR V2$="ZIW" THEN V
2$="LRW"
1450 IF V2$<>"GVI" AND V2$<>"LRW" THEN
 P$="GSV OZHVI HSLG SZH NL VUUVXG":GOT
O N9
1460 IF V2$="GVI" AND D3=N0 THEN P$="R
 WLM'G HVV ZMB NLMHGVI SVIV":GOTO N9
1470 IF V2$="LRW" AND D7=N0 AND E0=N0
AND E3=N0 AND E7=N0 THEN P$="R WLM'G H
VV ZMB ZMWILRWH SVIV":GOTO N9
1480 T=INT(100*RND(N0))+1:IF T>P2+P3+5
0 THEN P$="BLF URIV ZMW NRHH":GOTO N9
1490 IF D3<>N1 THEN 1500
1492 P$="BLF SRG GSV NLMHGVI":GOSUB N4
:D4=D4-((10+P2+P3)/N2):IF D4>N0 THEN 1
540
1494 D3=N0:D4=N0:P$="BLF SZEV PROOVW R
G":GOTO N9
1500 IF D7<>N1 THEN 1510
1502 P$="BLF SRG GSV ZMWILRW":GOSUB N4
:D8=D8-(5+P2+P2)/N2:IF D8<=N0 THEN D7=
N0:D8=N0:P$="RG RH WVHGILBVW":GOTO N9
1504 GOTO 1540
1510 IF E0<>N1 THEN 1520
1512 P$="BLF SRG GSV ZMWILRW":GOSUB N4
:E1=E1-(N5+P2+P3)/N2:IF E1<=N0 THEN E0
=N0:E1=N0:P$="RG RH WVHGILBVW":GOTO N9
1514 GOTO 1540
1520 IF E3<>N1 THEN 1530
1522 P$="BLF SRG GSV ZMWILRW":GOSUB N4
:E4=E4-(N5+P2+P3)/N2:IF E4<=N0 THEN E3
=N0:E4=N0:P$="RG RH WVHGILBVW":GOTO N9
1524 GOTO 1540
1530 IF E7<>N1 THEN 1540
1532 P$="BLF SRG GSV ZMWILRW":GOSUB N4
:E8=E8-(N5+P2+P3)/N2:IF E8<=N0 THEN E7
=N0:E8=N0:P$="RG RH WVHGILBVW":GOTO N9
1534 GOTO 1540
1540 IF D3=N1 THEN P$="RG RH HGROO ZOR
EV":GOTO N9
1550 P$="GSV ZMWILRW RH HGROO UFMXGRLM
RMT":GOTO N9
1560 IF V2=N0 THEN P$="R XZM'G WL GSZG
":GOTO N9
1570 IF I(V2)<>N0 THEN P$="R WLM'G SZE
V GSZG":GOTO N9
```

```
1580 IF V2<>N9 AND V2<>14 THEN P$="R X
ZM'G WL GSZG":GOTO N9
1590 IF (V2=N9 AND A=44) OR (V2=14 AND
 A=38) THEN P$="MLGSRMT SZKKVMH":GOSUB
 N4
1600 IF V2<>N9 OR A<>38 THEN 1610
1602 F4=35:P$="GSV XLNKFGVI IVKORVH: '
YZHV WVHGIFXG  HVJFVMXV HGZIGVW'":GOSU
B N4:P$="WVHGIFXGRLM RM: "
1604 P$(LEN(P$)+N1)=STR$(F4):P$(LEN(P$
)+N1)=" NRMFGVH":P4=P4-N1:I(9)=100:GOT
O N9
1610 IF V2=14 AND A=45 THEN P$="GSV TZ
GV LKVMH":E9=N1:GOTO N9
1620 IF V2<>10 THEN P$="WLM'G YV IRWRX
FOLFH":GOTO N9
1630 IF I(10)<>NO THEN P$="BLF WLM'G S
ZEV GSZG":GOTO N9
1640 PRINT OK$:I(10)=50:P4=P4-N1:P1=P1
+N5+P3:IF P0<P1 THEN P0=P1
1650 GOTO L1
1660 IF A=N1 AND V2$="OFV" THEN P$="GS
V ZRIOLXP LKVMH... BLF ZIV YOLDM LFGRM
GL GSV EZXFFN LU HKZXV":GOTO N8
1670 IF A=6 AND V2$="OFV" THEN P$="Z H
GIZMTV, LIZMTV TOLD XLEVIH BLF ZMW GSV
M UZWVH ZDZB":GOTO N9
1680 IF A=10 AND V2$="IVU" AND D3=N1 T
HEN P$="MLGSRMT SZKKVMH":GOTO N9
1690 IF A=10 AND V2$="IVW" THEN D3=N1:
P$="ZM ZORVM NLMHGVI RH IVOZHVW.  RG R
H   ZGGZXPRMT BLF":GOTO N9
1700 IF A=13 AND V2$="OFV" THEN A=34:P
$="Z UOZHS LU ORTSG GVNKLIZIROB YORMWH
   BLF":GOTO N9
1710 IF A=20 AND V2$="IVW" AND D6=NO T
HEN P$="MLGSRMT SZKKVMH":GOTO N9
1720 IF A=20 AND V2$="IVW" THEN D6=NO:
P$="GSV HXIVVM TLVH YOZMP":GOTO N9
1730 IF A=20 AND V2$="OFV" THEN D6=N1:
P$="GSV HXIVVM ORTSGH FK":GOTO N9
1740 IF A=31 AND V2$="OFV" THEN E5=N1:
PRINT OK$:GOTO L1
1750 IF A=34 AND V2$="IVW" THEN A=13:P
$="Z UOZHS LU ORTSG GVNKLIZIROB YORMWH
   BLF":GOTO N9
1760 IF A=40 AND V2$="IVW" THEN F3=NO:
PRINT OK$:GOTO L1
1770 IF A=40 AND V2$="OFV" THEN F3=N1:
PRINT OK$:GOTO L1
1780 P$="MLGSRMT SZKKVMH":GOTO N9
1790 IF A=22 AND V2$="VVM" THEN P$="BL
F XZM HVV MLGSRMT LU RMGVIVHG LM GSVIZ
WZI":GOTO N9
1800 IF V2=NO THEN P$="R WLM'G SZEV GS
ZG":GOTO N9
1810 IF I(V2)<>NO AND A<>I(V2) THEN P$
="R WLM'G SZEV GSZG":GOTO N9
1820 IF V2=N3 OR V2=13 THEN P$="R HVV
MLGSRMT HKVXRZO":GOTO N9
1830 IF V2=N9 THEN P$="HLIIB, LMOB Z X
LNKFGVI XZM IVZW Z     KILTIZN":GOTO N
9
1840 IF V2=16 THEN P$="GSV KOZMH ZIV H
VZOVW...LMOB XLNNZMW   XZM LKVM GSVN":
GOTO N9
1850 P$="R XZM'G IVZW GSZG":GOTO N9
1860 GRAPHICS NO:P$="* * * KOZBVI'H RM
EVMGLIB * * *":GOSUB N4:PRINT
1870 FOR T=N1 TO 16:IF I(T)=NO THEN P$
=I$(IP(T),IP(T+N1)-N1):GOSUB N4
1880 NEXT T
1890 CLOSE #N1:OPEN #N1,N4,NO,"K:":GET
 #N1,T:CLOSE #N1
1900 GOTO 2410
1910 IF V2=NO THEN P$="R XZM'G WL GSZG
":GOTO N9
1920 IF I(V2)<>NO THEN P$="R WLM'G SZE
V GSZG":GOTO N9
1930 IF V2<>N1 OR A<>12 OR D5<>NO THEN
 1940
1932 P$="GSV XZYRMVG RH WVHGILBVW":D5=
N1:I(N1)=100:I(N5)=ABS(I(N5)):P4=P4-N1
:GOTO N9
1940 IF (V2=N1 OR V2=15) AND (D3=N1 OR
 D7=N1 OR E0=N1 OR E3=N1 OR E7=N1) THE
N I(V2)=100:P4=P4-100:GOTO 1490
1950 IF (V2<>N1 AND V2<>15) OR A<>N1 T
HEN 1960
```

```
1952 P$="GSV ZRIOLXP RH WVHGILBVW...BL
F ZIV    YOLDM LFG RMGL GSV EZXFFM LU
HKZXV":GOTO N8
1960 IF (V2<>N1 AND V2<>15) OR A<>36 O
R E6<>N0 OR E5<>N0 THEN 1970
1962 P$="GSV WLLI RH WVHGILBVW...BLF Z
IV YOLDM RMGL GSV FMKIVHHFIRAVW IVZXGL
I        YFROWRMT":GOTO N8
1970 IF (V2<>N1 AND V2<>15) OR A<>36 O
R E6<>N0 OR F3<>N0 THEN 1980
1972 P$="GSV WLLI RH WVHGILBVW.  BLF Z
IV       YLNYZIWVW DRGS IZWRZGRLM":GOT
O N8
1980 IF (V2=N1 OR V2=15) AND A=36 AND
E6=N0 THEN P$="GSV WLLI RH WVHGILBVW":
E6=N1:I(V2)=100:P4=P4-N1:GOTO N9
1990 IF V2<>N1 AND V2<>15 THEN 2000
1992 P$="GSV ":P$(5)=I$(IP(V2),IP(V2+N
1)-N1):GOSUB N4:P$="SZH ML ZUUVXG":I(V
2)=100:P4=P4-N1:GOTO N9
2000 GOTO 1370
2010 IF V2=N0 THEN P$="R XZM'G WL GSZG
":GOTO N9
2020 IF I(V2)<>N0 THEN P$="R WLM'G SZE
V GSZG":GOTO N9
2030 IF V2=N5 AND D3=N1 THEN V2$="GVI"
2040 IF V2=N5 AND (D7=N1 OR E0=N1 OR E
3=N1 OR E7=N1) THEN V2$="LRW"
2050 IF V2=N5 THEN 1390
2060 IF V2=N4 AND F3=N0 AND (A=40 OR A
=35 OR A=30 OR A=31) THEN P$="GSV YOZX
P WVERXV RH YORMPRMT":GOTO N9
2070 IF V2=N4 AND F3=N0 AND A=36 THEN
P$="GSV YOZXP WVERXV RH UOZHSRMT YIRTS
GOB":GOTO N9
2080 IF V2=N4 THEN P$="GSVIV ZIVM'G ZM
B ERHRYOV XLMGILOH LM  GSRH WVERXV":GO
TO N9
2090 IF V2<>12 THEN 2100
2092 I(12)=A:P4=P4-N1:I$(IP(12),IP(12)
+N4)="ZINVW":F2=35:P$="GSV YLNY DROO V
CKOLWV RM 35 NRMFGVH":GOTO N9
2100 IF V2=N2 AND A=12 AND D5=N0 THEN
D5=N1:P$="GSV XZYRMVG RH MLD LKVM":I(N
5)=ABS(I(N5)):GOTO N9
2110 IF V2=N2 AND A=12 AND D5=N1 THEN
P$="GSV XZYRMVG RH ZOIVZWB LKVM":GOTO
N9
2120 IF V2=7 AND A=26 AND D9=N0 THEN D
9=N1:I(16)=ABS(I(16)):P$="GSV HZUV LKV
MH":GOTO N9
2130 IF V2=11 THEN 2140
2132 P$="DSZG WL BLF DZMG NV GL WL DRG
S GSV":GOSUB N4:P$=I$(IP(V2),IP(V2+N1)
-N1):P$(LEN(P$)+N1)="?":GOTO N9
2140 IF I(N8)<>N0 THEN P$="GSVIV ZIVM'
G ZMB YZGGVIRVH ULI GSV IZWRL":GOTO N9
2150 IF F2=N0 THEN 2160
2152 P$="Z ELRXV HZBH 'YLNY HGZGFH: ":
P$(LEN(P$)+1)=STR$(F2):P$(LEN(P$)+1)="
 NRMFGVH":GOSUB N4
2154 P$="FMGRO WVGLMZGRLM'":GOTO N9
2160 P$="GSV IZWRL RH HROVMG":GOTO N9
2170 GRAPHICS N0:P$="* * * KOZBVI'H HG
ZGFH * * *":GOSUB N4:PRINT
2180 P$="XFIIVMG SRG KLRMGH = ":P$(LEN
(P$)+N1)=STR$(P1):GOSUB N4
2190 P$="WVCGVIRGB ZGGIRYFGV= ":P$(LEN
(P$)+N1)=STR$(P2):GOSUB N4
2200 P$="OFXP ZGGIRYFGV     = ":P$(LEN
(P$)+N1)=STR$(P3):GOSUB N4
2210 REM
2220 CLOSE #N1:OPEN #N1,N4,N0,"K:":GET
 #N1,T:CLOSE #N1
2230 REM
2240 GOTO 2410
2250 GRAPHICS N0:P$="GAME OVER":GOSUB
N4:GOTO 2530
```

Update player status. Conduct compbat if appropriate.

```
2260 IF F2<>N0 THEN F2=F2-N1:IF F2<=N0
 THEN 2440
2270 IF F4<>N0 THEN F4=F4-N1:IF F4<=N0
 THEN 2480
2280 IF P1<P0 THEN P5=P5+0.5:IF P5=N1
THEN P5=N0:P1=P1+N1
2290 IF D3=N0 AND D7=N0 AND E0=N0 AND
E3=N0 AND E7=N0 THEN 2400
2300 T=INT(RND(N0)*100+N1):PRINT
```

```
2310 IF D3=N1 THEN P$="GSV NLMHGVI ZGG
ZXPH":GOSUB N4
2320 IF D3<>N1 THEN P$="GSV HVXFIRGB Z
MWILRW HSLLGH...":GOSUB N4
2330 IF T>80-(P2+P3) THEN P$="RG NRHHV
W":GOSUB N4:GOTO 2400
2340 P1=P1-INT(RND(NO)*N5)-INT(RND(NO)
*N5)-INT(RND(NO)*N5)-INT(RND(NO)*N5)-2
4+P3
2342 IF D3<>N1 THEN P1=P1+N5
2344 IF P1<NO THEN 2520
2390 P$="BLF ZIV SRG":GOSUB N4
2400 IF V1=0 OR V1>4 OR D3+D7+E0+E3+E7
>0 THEN FOR ZZ=N1 TO 150:NEXT ZZ
```

Initialize for new turn. Jump to appropriate room description.

```
2410 V0$="":V$="":V1$="":V2$="":V1=NO:
V2=NO:A$="":B$="":C$="":D$="":E$="":N=
NO:S=NO:W=NO:E=NO
2420 IF A>45 THEN 2560
2422 GOSUB A*10:GOTO 470
```

Evaluate end-game conditions and display appropriate messages.

```
2440 GRAPHICS NO:IF A=46 THEN F2=-1:GO
TO 2560
2450 P$="GSV KSLGLM YLNY VCKOLWVH...GS
V VMGRIV XLNKOVC RH WVHGILBVW":GOSUB N
4
2460 P$="BLF SZEV YVVM PROOVW YB GSV U
LIXV LU  GSV YOZHG":GOSUB N4
2470 PRINT :PRINT :GOTO 2530
2480 GRAPHICS NO:IF A=46 THEN F4=-1:GO
TO 2560
2490 IF A=38 THEN P$="GSV XLNKFGVI UOZ
HSVH YIRTSGOB,        VNRGGRMT HKZIPH
RM ZOO WRIVXGRLMH":GOSUB N4
2500 P$="GSV XLNKOVC HFWWVMOB VCKOLWVH
 RMGL    NROORLMH LU KRVXVH":GOSUB N4
2510 P$="BLF ZIV PROOVW YB GSV UZOORMT
 WVYIRH  ZILFMW BLF":GOSUB N4:PRINT :P
RINT :GOTO 2530
2520 PRINT :P$="BLF ZIV WVZW!":GOSUB N
4
2530 PRINT :PRINT "DO YOU WANT TO PLAY
 AGAIN";:INPUT A$
2540 IF A$(N1,N1)="B" THEN RUN
2550 GRAPHICS NO:END
2560 P$="GSV HKZXV HSRK HFWWVMOB ORUGH
 RMGL    LIYRG ZILFMW GSV KOZMVG":GOSU
B N4
2570 IF (F2=NO OR (F2<>NO AND I(12)<>4
1)) AND F4=NO THEN 2574
2572 GOTO 2580
2574 P$="BLF WRWM'G WVHGILB GSV YZHV.
 BLF SZEVUZROVW BLFI NRHHRLM.":GOSUB N
4:PRINT :GOTO 2530
2580 P$="UILN Z WRHGZMXV, BLF XZM HVV
GSV ZORVMYZHV VCKOLWV":GOSUB N4
2590 IF I(16)<>NO THEN P$="BLF WRWM'G
IVXLEVI GSV HVXIVG KOZMH   MVVWVW YB H
GZI XLNNZMW":GOSUB N4:PRINT :GOTO 2530
2600 P$="NRHHRLM RH Z HFXXVHH!":GOSUB
N4
2610 END
```

Initialize workspace. Read in items and verbs.

```
2620 CLR :DIM I$(300),I(16),IP(17),VB$
(51)
2622 DIM V$(20),V1$(3),V2$(3),A$(120),
B$(70),C$(40),D$(40),E$(40),P$(120)
2624 DIM Z$(40),OK$(3),V0$(20)
2626 NO=0:N1=1:N2=2:N3=3:N4=4:N5=5:N8=
8:N9=9:L1=2260:L2=1110:L3=1100:L4=1120
:OK$="OK":OK$(N3)=CHR$(253)
2630 GRAPHICS NO
2640 PRINT "  OPERATION:  SABOTAGE BY
 RAY SATO":? :? "    ATARI VERSION BY
RICH BOUCHARD"
2650 I$="":FOR T=N1 TO 16:READ Z$:IP(T
)=LEN(I$)+1:I$(LEN(I$)+N1)=Z$:READ Z:I
(T)=Z
2660 NEXT T:IP(T)=LEN(I$)+1
2670 READ VB$
```

Establish player-attribute points. Jump to first room.

```
2700 P0=40:FOR T=N1 TO P0:P0=P0+INT(RN
D(NO)*N2):NEXT T
```

```
2710 P1=P0
2720 P2=10:FOR T=N1 TO P2:P2=P2+INT(RN
D(N0)*N2):NEXT T
2730 P3=10:FOR T=N1 TO P3:P3=P3+INT(RN
D(N0)*N2):NEXT T
2740 D4=50:D8=D4:E1=D4:E4=D4:E8=D4:FOR
 T=N1 TO D4:D4=D4+INT(RND(N0)*N2):D8=D
8+INT(RND(N0)*N2)
2742 E1=E1+INT(RND(N0)*N2):E4=E4+INT(R
ND(N0)*N2):E8=E8+INT(RND(N0)*N2):NEXT
T
2750 A=N1:P4=N1
2760 GOTO 2410
```

Item and verb data.

```
2770 DATA KOZHGRX VCKOLHREV,0,XILDYZI,
7,XZOVMWZI,8,HNZOO YOZXP WVERXV,9,OZHV
I KRHGLO,-12,HVXFIRGB PVB,-16
2772 DATA VOVXGILMRX XLMGILO YZGLM,-17
,YZGGVIRVH,18,XLNKFGVI WVHGIFXG KILTIZ
N,21,HROEVI KROO,23
2774 DATA KLIGZYOV IZWRL,25,OZITV KSLG
LM YLNY,28
2780 DATA TZOZXGRX XSZIG,32,OZFMXS HBH
GVN XZHHVGGV,-32,MRGILTOBXVIRM,39,HVXI
VG KOZMH,-26
2790 DATA MLIHLFDVHVZHLKVTVGWILHSLRMHV
ZGKFHIVZRMEGSIFHVHGZJFR
```

LEYTE

by Victor A. Vernon Jr.

Atari version by Alan J. Zett

***Leyte* is a sea-battle simulation for an Atari with 16K RAM.**

After the U.S. Central Pacific (CINCPAC) forces under Admiral Nimitz had taken the islands of Pelelu and Angour, and MacArthur's forces had secured New Guinea, President Roosevelt met with MacArthur and Nimitz. MacArthur convinced the President that the next objective ought to be the liberation of the Philippines, thereby tearing the Japanese empire in two, isolating the home islands from the oil and wealth of the Indies.

The American plan was to invade the island of Leyte in the center of the Phillipines with the combined forces of MacArthur's Eighth Army and Admiral Kinkaid's Seventh Fleet, with Admiral Halsey's Third Fleet protecting the landing attempt from disruptions from the Imperial Fleet. This would mark the first joint operation by the two American Pacific Fleets.

The Seventh Fleet was mainly an invasion fleet, consisting of troop transports and LST's. Its only capital ships were in a bombardment group of old battleships (some of them veterans of Pearl Harbor), a few cruisers, and destroyers with escorts. This group was not prepared for large-scale naval engagements, and lacked both torpedoes and armor-piercing bombs.

Fortunately, the Third Fleet was much better equipped, with its new Essex class carriers, each with 100 aircraft, and its several lighter carriers, Iowa class battleships, heavy and light cruisers, and destroyers. The combination of the two Fleets was the greatest armada ever assembled.

The Japanese, woefully outclassed, developed a plan, known as SHO-1, intended to defend the core of their Empire and win an agreement that would preserve as much of it as possible. Their strategy was to draw the Americans out on a limb, then cut off the limb. With the Leyte landings, they were prepared for that surgery.

SHO-1 was founded upon two principles: 1) The Imperial Fleet was no match for the Third Fleet; 2) The Seventh Fleet was no match for the Imperial Fleet.

The Japanese would therefore have to neutralize the Third Fleet, leaving the Seventh at the mercy of the Imperial Fleet. This would require perfect timing and fabulous luck. Amazingly, SHO-1 succeeded in almost every detail. Only an error of judgement denied the Japanese full victory.

The Japanese forces were divided into three fleets: the Northern, the Central, and the Southern. The Northern Fleet's mission was to lure Admiral Halsey and the Third north, away from Leyte, while the Central and Southern

Fleets drove the Americans out of the Philippines.

The Southern force was to sail through the Suriago Strait, which led to the southern portion of Leyte Gulf. The Central force, under Admiral Kurita, was to be the main attack force. It would sail through the Sibuyan Sea, through the San Bernardino Strait, then turn south around the island of Samar to enter Leyte Gulf from the north. The Central and Northern forces were to meet early on October 25, 1944, and together wreak havoc on the American transports in the harbor.

Let us now briefly review the events of October 20 through 25. Remember that this was the largest naval battle ever fought; it covered thousands of square miles of ocean, and involved every type of warship, from the smallest PT boat to the largest battleship. If this brief account proves confusing, you might wish to refer to Edwin P. Hoyt's *The Battle of Leyte Gulf* and Admiral Morrison's *Naval Operations in World War II.*

On October 20, the Americans landed on the western shore of Leyte Island. Almost immediately, SHO-1 was put into operation. The landings were accomplished with only some Japanese air attacks based from Luzon. Two days later, American submarines spotted the Japanese Central force in the Sulu Sea. At first light, Halsey sent out his search planes. They found the Japanese in the Subayan Sea, sailing east toward San Bernardino Strait. Halsey immediately launched a full air strike.

About this time, Halsey ordered the formation of a new task group, under Admiral Lee, whose job it was to guard the San Bernardino Strait if and when Halsey found the Japanese carriers. Meanwhile, The Japanese Northern force was trying desperately to be found, but was having little success. Although they had broken radio silence, atmospheric conditions prevented Halsey's radio men from detecting them.

Halsey's air strike found its target, and, after inflicting some damage, misinterpreted a Japanese withdrawal as a full retreat. By now, the Northern force had been discovered, and Halsey decided to attack them. He took his special task force, which had been guarding the strait, with him, but a confusion occurred that left the Seventh Fleet's commander, Admiral Kinkaid, with the impression that the strait was still guarded. As a result, Kinkaid deployed no forces to the strait, hoping to concentrate all his efforts against the Japanese Southern force.

The night of October 24-25 was most eventful. Halsey and the Northern force were approaching one another. The Central force had turned east, and was heading for the mouth of San Bernardino Strait. The Southern force was sailing north through Surigoa into a trap, laid by Admiral Kinkaid, and was utterly destroyed.

October 25 was a crucial day in the Pacific War. San Bernardino Strait was an unencumbered entrance into Leyte Gulf, and the Central force was steaming through it at 20 knots. Should these 22 ships reach the landings, they would transform them into a bloody shambles, with American help too far to the north and south.

The thin American defenses in the strait sacrificed themselves. Their reckless, desperate attacks, though continually smashed, served to scatter and confuse the Japanese. Kurita, the commander of the Central force, knew nothing of the Northern's success in luring the Third Fleet away. He thought that he was driving the last remnants of the Imperial Fleet into a deadly trap. In his confusion, he thought that the few, light ships defending Leyte were the heavy ships of the Third Fleet. Though he could have easily smashed the defenders, he turned around. In the next few days, Halsey's airmen smashed Kurita's Central force. Although SHO-1 had worked perfectly, all three divisions of the Imperial Fleet were defeated. They lost 26 of their finest ships in four short days.

What *might* have happened if Lee's task force had been awaiting Kurita's Central force at the mouth of San Bernardino Strait? In this simulation, you will be Lee, and the computer, Kurita. Your fleet will consist of the battleships Jersey, Washington, Massachussetts, and Alabama; the cruisers Pittsburgh and Baltimore; and six Fletcher class destroyers. The Japanese fleet will be the battleships Yamato, Nagato, Kongo, and Hagura; the heavy cruisers Chokai and Sizuya; the light cruiser Nagara; and eight destroyers.

In both fleets, the destroyers are considered to be a collective force. That is, the target for one destroyer is the target for all. While individual destroyers may be targeted and sunk, all of them must be sunk in order to destroy their force.

The screen displays a map showing the Island of Samar, with San Bernardino to the left and Leyte Gulf to the right. North is the left side of the screen; east is the top; south is right; and west is the bottom. The Japanese will move from left to right, from San Bernardino to Leyte. To win, you must sink all seven capital ships — everything larger than a destroyer. If any get to Leyte, or if all your ships (including your destroyers) are sunk. Of course, the Japanese will *not* turn around.

You can issue any of these four commands by pressing the appropriate number key:

1 - status report
2 - fire guns
3 - fire torpedoes
4 - course correction

The status report will tell you the condition of each of your ships; you will not be told the condition of the Japanese ships. (The computer will tell you only if they are sunk, at the time when it checks the status of all ships.)

If you choose to fire your guns you will then be asked for a target for each of your ships. The Japanese ships are numbered as follows; use the number to designate a target:

1 - Yamato
2 - Nagota
3 - Kongo
4 - Haguro
5 - Chokai
6 - Suzuya
7 - Nagara
8 - Destroyers (collectively)

If a target is sunk you will be directed to select another. Whether or not you score a hit is determined by three factors:

1 - a randomly selected number
2 - how close you are to the target
3 - whether the target has been fired upon by another ship.

The closer you are to the Japanese, the better will be your chances to hit something. The third factor comes into play because naval gunfire is accomplished by first firing a round, and then correcting the aim by noting where the first shells land. This can be done by watching the splashes; but if two or more ships are firing on the same target, it is impossible to tell which splashes go with each ship's fire. The Japanese avoided this by coloring their shells, thus giving different colored splashes for each ship.

Firing torpedoes is essentially the same, except that only destroyers carry torpedoes; thus, you will only be able to fire at one target. Also remember that while a torpedo can do much more damage than a shell, the Mark 14 torpedoes used by the U.S. Navy in World War II were notoriously unreliable. At times they would sink; at other times they would breach and explode against a wave. If they did hit a ship, they didn't always explode. The Japanese "long Lance" torpedo was larger (24 inches versus 21), and it *worked.*

Command #4 is a course correction; it is not necessary to access this command on every turn. Your course is initially set in a northerly direction. To change this you will be prompted to enter a number which will lead you in any one of 8 directions. The Japanese fleet will be heading generally south at a flank speed, so that if they get past your fleet, you will not be able to intercept them. Also remember that you are a Fleet Commander, not a Ship's Captain. Your orders will be for the fleet as a whole, and you will not be able to move individual ships.

Basically that's all there is to it: maneuver and fire. But remember that maneuvering is as important as firing. If you stay too far away from the Japanese, your chances of sinking anything will be very slim, and they will sail on to Leyte while you watch them go by. But if you get too close too soon, the Japanese will blow you out of the water with their superior gunfire and torpedoes (Japanese cruisers also carry torpedoes).

Your position on the map will be indicated by an "A," and the Japanese position by a "J." If both fleets are at the same place you will see an "*".

Variables

A: Location of American fleet.
A(n,nn): Holds all pertinent information about the American fleet.
A$(n): Names of American ships.
A1: Course of American fleet.
AZ: Miscellaneous.
C: Movement counter. Determines when computer will check on status of all ships. Also helps determine if Japanese fire guns or torpedoes.
G: Command variable.
H: Hit counter.
H(n,nn) or H(n,nn,x): Hit table; determines how much damage is done if a hit is scored. Damage is determined by the size of the shell, where it hits, and what it hit.
I: Used in POKEing and PEEKing screen memory.
I$: Keyboard input variable.
J: Position of Japanese fleet.
J(n,nn): Holds all pertinent information about the Japanese fleet.
J$(n): Names of Japanese ships.
JT: Number of reserve torpedoes for Japanese fleet.
M: Modifier; changes range of Z, depending upon fleet locations.
M1: Temporary storage for M.
NT: Number of reserve torpedoes for American fleet.
R$: Used in status report messages.
T: Used in arrays to signify target.
T$: Keyboard input variable.
T1,T2: Temporary storage variables.
T1$,T2$: Temporary storage variables

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS           'Leyte'           SS
SS  Author: Victor Vernon Jr.  SS
SS  Translator: Alan J. Zett   SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Initialization.

```
50 DIM A(7,7),J(7,7),A$(120),J$(120),H
(4,12),T1$(40),T2$(15),R$(40)
```

Load the arrays.

```
60 GRAPHICS 0:I=PEEK(560)+PEEK(561)*25
6+4:I=PEEK(I)+PEEK(I+1)*256
70 POKE 752,1:? "ONE MOMENT PLEASE":?
"THERE IS A LOT OF DATA TO READ":NT=50
:JT=100:A=167:A1=-1:POKE 82,0
80 FOR X=1 TO 105:J$(X)=" ":A$(X)=" ":
NEXT X:RESTORE
```

Draw the map on the screen. Map cannot be redrawn during the program.

```
100 FOR X=0 TO 7:READ T1$,T2$:J$(X*15+
1)=T1$:A$(X*15+1)=T2$:FOR Y=0 TO 7:REA
```

```
D T1,T2:J(X,Y)=T1:A(X,Y)=T2:NEXT Y:NEX
T X
110 FOR X=0 TO 4:FOR Y=1 TO 6:FOR Z=0
TO 1:READ T1:H(X,Y+Z*6)=T1:NEXT Z:NEXT
 Y:NEXT X
113 FOR X=1 TO 40:T1$(X)=CHR$(160):NEX
T X:OPEN #1,4,0,"K"
```

Move the fleets.

```
115 GRAPHICS 0:POKE 752,1:FOR X=1 TO 1
3:? ",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,";:NEXT X
120 POSITION 9,7:? T1$(21);:POSITION 7
,8:? T1$(17);:POSITION 5,9:? T1$(13);:
POSITION 4,10:? T1$(11);
125 POSITION 1,11:? T1$(6);:POSITION 0
,12:? T1$(2);:POSITION 0,13:PRINT T1$;
T1$;
127 POSITION 0,13:? " < SAN BERNARDINO
 STR.      LEYTE GULF >";
130 POKE I+371,51:POKE I+375,33:POKE I
+379,45:POKE I+383,33:POKE I+387,50
140 POKE 82,2
```

Command routine.

```
150 FOR X=1 TO 1:NEXT X:C=0:H=0
160 POSITION 2,16:GOSUB 7000:POKE I+J,
12:POKE I+A,12
165 READ J:IF J=999 THEN 5010
170 M1=3:Z=INT(RND(0)*3)+1:IF Z=1 THEN
 J=J+40
172 IF Z=2 THEN J=J-40
175 IF PEEK(I+A+A1)<>128 THEN 177
176 POSITION 2,16:? "YOU'RE ABOUT TO R
UN AGROUND!":? "I'LL CORRECT YOUR POSI
TION.":FOR ZZ=0 TO 500:NEXT ZZ:A1=-40
177 IF A+A1<0 THEN A1=40
180 A=A+A1:POKE I+A,161:POKE I+J,170:I
F A=J THEN POKE I+A,138
```

Display the status of the American ships.

```
190 POSITION 2,16:GOSUB 7000:? "COMMAN
D ?":IF J=A+1 OR J=A-1 OR J=A+40 OR J=
A-40 THEN M1=0
200 POSITION 2,18:? " 1 - STATUS REPOR
T":IF A=J THEN M1=-1
210 ? " 2 - FIRE GUNS":IF A=J+39 OR A=
J-39 OR A=J+41 OR A=J-41 THEN M1=2
220 ? " 3 - FIRE TORPEDOES":IF A=J+1 O
R A=J-1 OR A=J+40 OR A=J-40 THEN M1=0
230 ? " 4 - COURSE CORRECTION":M=M1
240 POKE 764,255:FOR X=1 TO 2000:IF PE
EK(764)=255 THEN 270
250 GET #1,T1:IF T1<49 OR T1>52 THEN 2
70
255 FOR X=1 TO 1:NEXT X
260 G=VAL(CHR$(T1)):ON G GOTO 280,480,
620,730
270 NEXT X:C=C+1:IF C>2 THEN 150
275 GOTO 240
```

Changes half-turn count, and skips any sunken ships.

```
280 FOR X=0 TO 7:IF A(X,5)<1 THEN R$="
SUNK":GOTO 430
290 IF X>1 THEN 340
300 IF A(X,4)<50 OR A(X,2)<7 THEN R$="
FLOATING JUNK YARD":GOTO 430
310 IF A(X,4)<100 OR A(X,2)<12 THEN R$
="VERY HEAVY DAMAGE":GOTO 430
320 IF A(X,4)<150 OR A(X,2)<17 THEN R$
="MODERATE DAMAGE":GOTO 430
330 R$="ESSENTIALLY UNDAMAGED":GOTO 43
0
340 IF X>4 THEN 380
350 IF A(X,4)<50 OR A(X,2)<7 THEN R$="
PREPARE TO ABANDON SHIP!":GOTO 430
360 IF A(X,4)<100 OR A(X,2)<14 THEN R$
="SEVERE DAMAGE":GOTO 430
370 R$="LITTLE OR NO DAMAGE":GOTO 430
380 IF X=4 THEN 420
390 IF A(X,4)<20 OR A(X,2)<4 THEN R$="
SINKING!":GOTO 430
400 IF A(X,4)<50 OR A(X,2)<8 THEN R$="
HEAVY DAMAGE!":GOTO 430
410 R$="LIGHT OR NO DAMAGE":GOTO 430
420 IF A(X,4)<25 OR A(X,2)<5 THEN R$="
HEAVY DAMAGE!":GOTO 430
425 R$="UNDAMAGED"
```

```
430 POSITION 2,16:GOSUB 7000:? "STATUS
:"
440 ? A$(X*15+1,(X+1)*15):? R$:IF X=7
AND A(7,5)>1 THEN ? A(7,5);" AFLOAT"
460 FOR Y=0 TO 500:NEXT Y:NEXT X
470 GOTO 190
```

Prompt for a target. Reject any non-numeric input.

```
480 C=C+1:FOR X=0 TO 7:IF A(X,5)<1 THE
N NEXT X
```

Array is subscripted 0-7, so subtract 1 from the target. Also, change modifier to make target harder to hit if ship was was already fired upon this turn.

```
490 POSITION 2,16:GOSUB 7000:? "ENTER
TARGET FOR ";A$(X*15+1,(X+1)*15):M=M1
500 GET #1,T1:T=VAL(CHR$(T1))
510 IF T<1 OR T>8 THEN POSITION 2,17:?
 "ENTER NUMBER OF TARGET SHIP":? "CHEC
K INSTRUCTIONS":GOTO 500
```

If target is sunk, select another.

```
520 T=T-1:M=M+J(T,7):J(T,7)=J(T,7)=J(T
,7)+0.5
```

Reset hit counter to zero.

```
530 IF J(T,5)<1 THEN ? J$(T*15+1,(T+1)
*15):? "WAS SUNK":FOR ZZ=0 TO 200:NEXT
 ZZ:GOTO 490
540 H=0:FOR Y=1 TO A(X,6):Z=INT((RND(0
)*10)+1)+M:IF Z>4 THEN 570
545 H=H+1
```

V determines where target was hit. If the ship firing was a battleship, and the target is hit on the front or back turret or on the bridge, then the number of guns on the target is decreased by one.

```
550 V=INT(RND(0)*5):IF X<5 AND (V=0 OR
 V=1 OR V=3) THEN J(T,6)=J(T,6)-1
```

Timing delay.

```
555 FOR Z=0 TO 200:NEXT Z
```

Change defense factors on target.

```
560 J(T,2)=J(T,2)-H(V,A(X,0)+6):J(T,4)
=J(T,4)-H(V,A(X,0))
```

Display the number of main armament hits.

```
570 NEXT Y:IF H>0 THEN POSITION 2,18:G
OSUB 7000:? H;" MAIN ARMAMENT HITS ON:
":? J$(T*15+1,(T+1)*15):FOR ZZ=1 TO 50
0:NEXT ZZ:H=0
```

Repeat the process for secondary armaments.

```
580 M=M1:FOR Y=1 TO A(X,6)*2:Z=INT((RN
D(0)*10)+1)+M:IF Z>4 THEN 610
590 H=H+1
600 V=INT(RND(0)*5):J(T,2)=J(T,2)-H(V,
A(X,1)+6):J(T,4)=J(T,4)-H(V,A(X,1))
610 NEXT Y:NEXT X:M=M1:IF H>0 THEN POS
ITION 2,19:GOSUB 7000:? H;" SECONDARY
ARMAMENTS HITS ON:":? J$(T*15+1,(T+1)*
15):H=0
615 FOR ZZ=0 TO 500:NEXT ZZ:GOTO 880
```

U.S. torpedo fire.

```
620 C=C+1:M=M1-C:POSITION 2,16:GOSUB 7
000
630 IF A(7,3)<1 AND NT<1 THEN ? "NO TO
RPEDOES TO FIRE":FOR ZZ=0 TO 500:NEXT
ZZ:GOTO 880
640 IF A(7,3)<1 AND NT>9 THEN A(7,3)=1
0:NT=NT-10
650 A(7,3)=A(7,3)-5:? "TARGET ? (ENTER
 NUMBER PLEASE)":M=M1
670 GET #1,T1:T=VAL(CHR$(T1)):IF T<1 O
R T>8 THEN ? "NO SUCH TARGET":GOTO 670
680 T=T-1:IF J(T,5)<1 THEN POSITION 2,
18:? J$(T*15+1,(T+1)*15):? "WAS SUNK":
? "SELECT ANOTHER TARGET":GOTO 670
690 FOR Y=1 TO 5:Z=INT((RND(0)*10)+1)+
M:IF Z>5 THEN 720
700 V=INT(RND(0)*5):J(T,2)=J(T,2)-H(V,
12):J(T,4)=J(T,4)-H(V,6):H=H+1
```

```
710 POSITION 2,18:? H;" HITS ON: ";J$(
T*15+1,(T+1)*15):GOSUB 7000:FOR ZZ=0 T
O 200:NEXT ZZ
720 NEXT Y:GOTO 880
```

U.S. course change routine.

```
730 POSITION 2,16:GOSUB 7000:? "   8
1  2":? "      !":? "   7--+--3   ENTE
R NEW COURSE"
740 ? "      !":? "   6  5  4"
760 GET #1,T1:G=VAL(CHR$(T1))
770 IF G<1 OR G>8 THEN 760
780 GOTO G*10+780
790 A1=-40:GOTO 870
800 A1=-39:GOTO 870
810 A1=1:GOTO 870
820 A1=41:GOTO 870
830 A1=40:GOTO 870
840 A1=39:GOTO 870
850 A1=-1:GOTO 870
860 A1=-41
870 GOTO 190
```

Check turn count. Japanese fire torpedoes on second half-turn if range is close enough.

```
880 IF C>1 AND M1<3 THEN 1020
885 M=M1:POSITION 2,16:GOSUB 7000:H=0
```

Japanese gunfire routine.

```
890 FOR Y=0 TO 7:IF J(Y,5)<1 THEN 1010
895 POSITION 2,16:? "INCOMING JAPANESE
 FIRE FROM:          ":GOSUB 7000:? J$(
Y*15+1,(Y+1)*15)
900 FOR X=0 TO 7:IF A(X,5)<1 THEN NEXT
 X:GOTO 4070
910 T=TT:FOR X=1 TO 1:NEXT X:TT=TT+1:I
F TT>7 THEN TT=0
915 IF A(T,5)<1 THEN 910
920 FOR X=1 TO J(Y,6):Z=INT((RND(0)*10
)+1)+M:IF Z>5 THEN 960
940 M=M+0.5:H=H+1:V=INT(RND(0)*5):IF (
V=1 OR V=3) AND Y<4 THEN A(T,6)=A(T,6)
-1
950 A(T,2)=A(T,2)-H(V,J(Y,0)+6):A(T,4)
=A(T,4)-H(V,J(Y,0))
960 NEXT X:IF H>0 THEN POSITION 2,19:?
 H;" MAIN ARMAMENT HITS ON:":GOSUB 700
0:? A$(T*15+1,(T+1)*15):FOR ZZ=0 TO 50
0:NEXT ZZ:H=0
965 IF Y>5 THEN 1010
970 FOR X=1 TO J(Y,7):Z=INT((RND(0)*10
)+1)+M:IF Z>5 THEN 1000
980 H=H+1:V=INT(RND(0)*5):A(T,2)=A(T,2
)-H(V,J(Y,1)+6):A(T,4)=A(T,4)-H(V,J(Y,
1))
1000 NEXT X:IF H>0 THEN POSITION 2,19:
GOSUB 7000:? H;" SECONDARY ARMAMENT HI
TS ON:":? A$(T*15+1,(T+1)*15):H=0:FOR
ZZ=0 TO 500:NEXT ZZ
1010 NEXT Y:FOR X=0 TO 100:NEXT X
1015 IF C>1 THEN 3020
1017 GOTO 160
```

Japanese torpedo fire routine.

```
1020 M=M1:POSITION 2,16:GOSUB 7000:? "
INCOMING JAPANESE TORPEDO FIRE":IF M1=
3 THEN M=M+1
1025 FOR ZZ=0 TO 300:NEXT ZZ
1030 FOR X=4 TO 6:IF J(X,3)>0 AND J(X,
5)>0 THEN 2040
1035 NEXT X
1040 IF (J(7,3)<1 AND JT<1) OR J(7,5)<
1 THEN 3020
1050 IF J(7,3)<1 AND JT>9 THEN J(7,3)=
10:JT=JT-10
1060 J(7,3)=J(7,3)-5
1070 FOR X=0 TO 7:IF A(X,5)<1 THEN NEX
T X:GOTO 4070
1080 T=TT:FOR X=1 TO 1:NEXT X:TT=TT+1:
IF TT>7 THEN TT=0
1085 IF A(T,5)<1 THEN 1080
1090 FOR X=1 TO 5:Z=INT((RND(0)*10)+1)
+M:IF Z>4 THEN 2030
1100 V=INT(RND(0)*5):A(T,2)=A(T,2)-H(V
,12):A(T,4)=A(T,4)-H(V,6):POSITION 2,1
8
1105 ? "TORPEDO HIT ON ";A$(T*15+1,(T+
1)*15):FOR ZZ=0 TO 500:NEXT ZZ:IF T<7
THEN T=T+1
1110 FOR ZZ=0 TO 500:NEXT ZZ
2030 NEXT X:GOTO 3020
```

```
2040 F=X:FOR X=1 TO 1:NEXT X:J(F,3)=J(
F,3)-8
2050 FOR X=0 TO 7:IF A(X,5)<1 THEN NEX
T X:GOTO 4070
2060 T=X:FOR X=1 TO 1:NEXT X:IF T=8 TH
EN 4070
2070 FOR X=1 TO 8:Z=INT((RND(0)*10)+1)
+M:IF Z>4 THEN 3010
2080 V=INT(RND(0)*5):A(T,2)=A(T,2)-H(V
,12):A(T,4)=A(T,4)-H(V,6):POSITION 2,1
8
2090 GOSUB 7000:? "TORPEDO HIT ON ";A$
(T*15+1,(T+1)*15):FOR ZZ=0 TO 500:NEXT
 ZZ:IF T<7 THEN T=T+1
3010 NEXT X
```

Status update.

```
3020 POSITION 2,16:GOSUB 7000:? "CHECK
ING FOR SUNKEN SHIPS":FOR ZZ=0 TO 400:
NEXT ZZ
3030 FOR X=0 TO 6:J(X,7)=0:IF J(X,2)<1
 OR J(X,4)<1 THEN J(X,5)=0:? J$(X*15+1
,(X+1)*15);"    ";:FOR Z=1 TO 500:NEXT
 Z
3050 IF A(X,2)<1 OR A(X,4)<1 THEN A(X,
5)=0
3060 NEXT X:J(7,7)=0:IF A(7,2)>0 OR A(
7,4)>0 THEN 3090
3070 A(7,5)=A(7,5)-1:IF A(7,5)<1 THEN
4020
3080 A(7,2)=5:A(7,4)=20:A(7,3)=10:NT=N
T-10
3090 IF J(7,2)>0 OR J(7,4)>0 THEN 4020
4010 J(7,2)=5:J(7,4)=25:J(7,3)=10:JT=J
T-10
4020 FOR X=0 TO 6:IF J(X,5)<1 THEN NEX
T X:GOTO 4030
4025 GOTO 4060
```

End-of-game messages.

```
4030 GRAPHICS 0:? "ALL JAPANESE CAPITO
L SHIPS SUNK":?
4040 ? "YOU HAVE SAVED THE LEYTE LANDI
NGS":? :?
4050 ? "TO PLAY AGAIN TYPE RUN":END
4060 FOR X=0 TO 7:IF A(X,5)>0 THEN 150
4070 NEXT X:GRAPHICS 0:? "YOU LOST ALL
 YOUR SHIPS":?
4080 ? "MAYBE KINKAID'S SEVENTH FLEET"
:?
4090 ? "WILL SAVE THE LEYTE LANDINGS":
? :? :GOTO 4050
5010 GRAPHICS 0:? "THE JAPANESE HAVE R
EACHED LEYTE":? :GOTO 4080
```

Data for all A and J arrays.

```
6000 DATA YAMATO,NEW JERSEY,1,2,4,4,35
,33,0,0,300,275,1,1,9,9,0,0
6010 DATA NAGATO,IOWA,2,2,4,4,30,33,0,
0,250,275,1,1,8,9,0,0
6020 DATA KONGO,WASHINGTON,3,2,4,5,29,
31,0,0,250,250,1,1,8,9,0,0
6030 DATA HAGURO,MASSACHUSETTS,3,2,4,5
,28,31,0,0,250,250,1,1,8,9,0,0
6040 DATA CHOKAI,ALABAMA,4,2,5,5,25,31
,16,0,220,250,1,1,8,9,0,0
6050 DATA SUZUYA,BALTIMORE,4,4,5,5,25,
25,16,0,220,200,1,1,15,9,0,0
6060 DATA NAGARA,PITTSBURGH,5,4,0,5,20
,25,8,0,100,150,1,1,7,9,0,0
6070 DATA JAP. DESTROYERS,U.S. DESTROY
ERS,5,5,0,5,10,15,10,10,50,55,8,6,6,5,
0,0
```

Data for hits table.

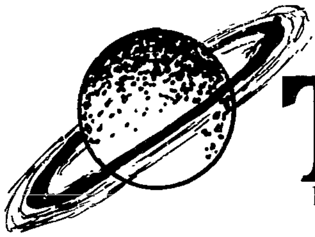
```
6080 DATA 20,2,15,1,12,1,8,0,2,0,20,1,
25,2,20,1,15,1,10,0,3,0,15,1
6090 DATA 18,1,15,1,12,0,7,0,2,0,17,3,
15,1,15,1,11,0,7,0,2,0,17,3
6100 DATA 17,2,15,2,12,2,10,1,3,1,20,5
```

Japanese movement data.

```
6110 DATA 361,362,323,284,285,247,208,
209,210,211,212,173,174,215,216,177,17
8,179,180,181,222
6120 DATA 183,224,185,226,227,188,229,
270,271,312,313,354,355,396,397,438,47
9,999,999
```

Erase to bottom of screen.

```
7000 ? "";:RETURN
7010 REM Line 7000:
     7 <ESC><SHIFT><DELETE>'S
```



TITAN

by William Morris and John Cope

***Titan* is an outer space mining simulation for a 24K Atari with one joystick.**

Editor's Note: This program is divided into two parts, enabling it to run in 24K RAM. Type in Part I, and SAVE it. Then type in Part II separately, and LIST it to disk or cassette with the LIST "D:filename" or LIST "C:" command.

The year is 2050. The Solar Mining Authority, in its ever-expanding quest for raw material, has finally decided to entertain bids from different corporations for the exclusive mining rights to Titan, one of the moons of Saturn.

In an attempt to ensure maximum efficiency from the operators of the Titan concession, the Solar Mining Authority has decided to permit up to four companies to operate on Titan using Probationary License Permits, for a period of one Earth year.

Sometime during the first quarter of the second year, an on-site inspection of the competing companies will result in one of them being awarded exclusive authority to mine the valuable Dilithium 3 crystals peculiar to Titan. The stakes are high; the risk and expense factors, even by 21st century standards, are enormous. The reward, however, makes the gamble more than worthwhile: for your company, fantastic profits; for you, possible promotion to the parent corporation's Board of Directors, as well as immense financial gain! Failure, of course, carries its own reward.

Having decided to accept the post of Superintendent of your corporation's Titan operation, you will be assigned one of four possible base sites: Actaeon, Bellona, Chimera, or Daedalus. As the highest ranking on-site representative of management, you must make the initial decisions concerning budgetary allotment. The only mandatory purchase is one power plant; all of the remaining choices are subject to your final authority. What about drilling rigs and robot miners? Drill rigs are required for vertical mining, while robominers carry out horizontal digging. Will you invest in an on-site Research and Development Station? Investing in this category of equipment can increase the likelihood of finding the Dilithium 3 crystals. How many Meteor Deflection Shields will be installed to protect your company's investment from the ever-present danger of stray asteroids from the rings? Refineries are needed, of course, to process the Dilithium 3 once you have discovered a vein of this elusive mineral. Without refineries, your profits will suffer greatly. Finally, energy consumption is a key factor on your Titan operation. Having an extra energizer unit or two could be very helpful! Remember: All of these factors are interrelated, with careful timing and meticulous attention to detail being necessary if the subsequent stages of your operation are to prosper. Costs fluctuate with increases or decreases of the availability of equipment.

Once you have made your initial budgetary decisions, you enter the management phase of your undertaking. Your decision concerning the number of surplus laborers (Labor

Pool), the length of their work shifts, the nature of the Recreational Facilities, as well as the nature of the Bonus Schedule and Safety Program, must tread the path best suited to achieve your goals. For example, cave-ins are a frequent problem on Titan because of gravity fluctutations caused by Saturn. Investing in safety measures could pay dividends. Your decision to pay your workers well will have positive results that are quickly evident. On the other hand, too much generosity in the area of labor relations could hamper the profit picture. The ultimate goal is to increase your efficiency; however, be advised that there are occasions when you will have to sacrifice efficiency for greater output. This knowledge can only be gained from "hands-on" experience. No one said the job was going to be easy!

Having set your policy for the current work cycle, your attention must now shift to the actual task of locating and mining the precious Dilithim 3 crystals. Once located at their designated mining strikes, your workers will await your decisions as to drilling locations. A unique screen controller will allow you to carefully monitor and direct the drilling phase and subsequent use of the robot miners. Through the careful interpretation of the Assay data beamed back to you, the number of drilling sites can be kept to a minimum. Again, the deployment of the robot miners will reflect your ability to carefully monitor on-site reports. Both of these latter utilities are prodigious users of power; but then, you will have thought of that when you made your initial decisions regarding power plant purchases or investment in Research and Development units. Right?

At the end of each work-cycle period you must decide, once again, on the budgetary decisions for the following month. Your experience during the previous cycle(s) should enable more efficient judgments as you progress. It is for this reason that you may view a summary of your decisions and accomplishments in the Monthly Report. Your statement will also reflect your financial capabilities for the next turn.

One minor last detail: As you will remember from an earlier part of this briefing, there is a periodic danger of damage to your operation from meteoroids. It will be your task to use the Meteor Deflection Shields to prevent any on-site damage to your corporation's property. Of course, these units require substantial amounts of power; but then, you have carefully planned for this contingency, haven't you?

Good luck and good mining!

On Playing Titan

One of the primary goals in designing Titan was to make it as independent of the keyboard as possible. For the Atari, this meant the use of the joystick.

Joystick Use

The joystick routine utilizes the up-down, left-right registers to permit viewing of possible choices and the button to register a final input. It is used in all of the key sections of the program.

Game Initialization

When indicating the number of players and the level of difficulty, simply use the left-right joystick capability to move the appropriate number and press the button to finalize your input.

Equipment Management and Labor Relations Phases

Toggling the joystick to the left or right indicates your decision to increase or decrease within a category, with the button registering this decision. Upward or downward action on the joystick moves you through the different categories.

Mining Phase

Left or right action on the joystick will allow you to move between the drill rig or robominer options, while upward toggling will permit you to consider ending this phase of the simulation. In each case pressing the joystick button registers your decision. Be aware, however, that the robominers cannot be activated unless you have at least one drill rig on site.

Once you have decided upon a drill rig or a robominer, you may move it to the drill site by using the joystick. Press the button when you have decided that you have the equipment on site. Downward toggling of the joystick permits drilling or the movement of the robominer down the shaft to occur. In the latter case you also have an upward movement capability within the drill shaft and lateral mining ability once the button has been pressed. Be aware that pressing the button during lateral mining terminates the robominer currently in use.

Monthly Report

Pressing the joystick button will terminate this section as you move on to decisions for the next month's mining operations.

Variables

A$: Name of player's base on Titan.
A: Drill Rig array.
AC: Action flag indicating decision to buy or sell.
C: Cost of items purchased during Equipment phase, or the price of worker-related decisions.
C2: Choice flag which is set during the Labor Management phase.
CH: Choice variable.
DA: Month of the year.
EF: Efficiency rating.
EN: Energy consumption.
EQ$: Equipment.
EX$: Extras.
FF: Efficiency counter.
HO: Drill hole(s).
J0-J7: Line numbers.
K: Player's variable table.
LV: Level of play.
P: Current player.
PL: Total number of players in simulation.
R, RA, RB: Random numbers.
UL, UN, UY, UZ: Sound variables for pitch, duration, etc.
V1: Horizontal location of dilithium.
V2: Vertical location of dilithium.
VE: Number of veins located.
WC$: Worker related items.
YE: Current year.

All other variables not specifically identified in the range of X-Z are counters or conditional flags.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS           'Titan'           SS
SS           Authors:          SS
SS William Morris & John Cope SS
SS      Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Initialization.

```
1100 DIM C(12),K(4,16),Z$(1),A$(40),EQ
$(70),WC$(70),EX$(40),DB$(72):J0=20000
:J1=20010:J2=20020:J3=20030:J4=20040
1110 J5=20050:J6=20060:J7=20070
1120 GRAPHICS 2:SETCOLOR 0,0,8:SETCOLO
R 1,0,8:SETCOLOR 2,8,0:SETCOLOR 4,8,0
1130 POSITION 7,4:FOR Z=0 TO 3:READ Z$
,Y,X:? #6;Z$;:SOUND Z,Y,10,4:FOR Y=1 T
O X:NEXT Y:NEXT Z:? #6;"N"
1140 SOUND 3,68,10,4:SOUND 0,136,10,4:
POKE 752,1:? "  (c) Wm. Morris & J. C
ope 1981":FOR Z=1 TO 999:NEXT Z
```

PLaying parameters.

```
1150 SOUND 3,0,0,0:FOR Z=0 TO 3:SOUND
0,162,10,8:SETCOLOR 2,3,15:SETCOLOR 4,
3,15:FOR Y=0 TO 200:NEXT Y
```

```
1160 SOUND 0,217,10,8:SETCOLOR 2,3,0:S
ETCOLOR 4,3,0:FOR Y=0 TO 200:NEXT Y:NE
XT Z
1170 FOR Z=0 TO 999:NEXT Z:FOR Z=0 TO
3:SOUND Z,0,0,0:NEXT Z:GOSUB 3000
1200 GRAPHICS 17:SETCOLOR 2,4,15:SETCO
LOR 4,12,0:POSITION 2,1:? #6;"WELCOME
TO TITAN!":POSITION 1,7:GOSUB J0
1210 ? #6;"USE THE JOYSTICK TO":? #6;"
SELECT THE NUMBER OF":POSITION 6,11:?
#6;"PLAYERS.":GOSUB 2100
1220 PL=X:GRAPHICS 17:SETCOLOR 2,4,15:
SETCOLOR 4,2,0:POSITION 0,7:? #6;"LEVE
L OF DIFFICULTY?":GOSUB 2100
1230 LV=X+4:GRAPHICS 17:SETCOLOR 4,15,
0
1240 FOR Z=1 TO PL:POSITION 1,4*Z-3:?
#6;"PLAYER # ";Z;" COMMANDS   BASE";A
$(Z*10-9,Z*10):GOSUB J1:NEXT Z
1250 POSITION 2,19:? #6;"PRESS TRIGGER
 TO":POSITION 8,21:? #6;"BEGIN":GOSUB
J0
1260 IF STRIG(0) THEN 1260
1270 FOR Z=1 TO PL:K(Z,6)=1:K(Z,13)=50
0:K(Z,14)=100:K(Z,15)=5:NEXT Z:? "}I'M
 NOW LOADING PART2."
1280 ? :? :? "GOTO300":POSITION 2,1:PO
KE 764,12:ENTER "D:TITAN32K.PT2"
```

Joystick selection routine.

```
2100 POSITION 6,17:Y=2110
2110 ? #6;"1 2 3 4":GOTO 2150
2120 ? #6;"1 2 3 4":GOTO 2150
2130 ? #6;"1 2 3 4":GOTO 2150
2140 ? #6;"1 2 3 4"
2150 GOSUB J1
2160 POSITION 6,17:FOR Z=0 TO 14 STEP
2:SETCOLOR 2,4,Z:NEXT Z:IF STRIG(0)=0
THEN X=(Y-2100)/10:GOSUB J0:RETURN
2170 IF STICK(0)=15 THEN 2160
2175 IF STICK(0)=11 THEN Y=Y-20:IF Y<2
100 THEN Y=2100
2180 Y=Y+10:IF Y>2140 THEN Y=2110
2190 GOTO Y
```

Set initial numeric variables and string arrays.

```
3000 DA=0:YE=2050:FOR Z=1 TO 4:FOR Y=0
 TO 16:K(Z,Y)=0:NEXT Y:NEXT Z
3010 C(1)=100:C(2)=200:C(3)=300:C(4)=4
00:C(5)=500:C(6)=2000:C(7)=5:C(8)=10:C
(9)=100:C(10)=200:C(11)=300:C(12)=400
3020 EQ$="DRILL RIG ROBOMINER DEFLECTO
R R&D UNIT  REFINERY  ENERGIZER CONTIN
UE  "
3030 A$=" ACTAEON   BELLONA   CHIMERA
  DAEDALUS  "
3040 WC$="LABOR POOLSHIFT TIMEWAGE SCA
LERECREATIONSAFETY    BONUSES   CONTIN
UE  "
3050 EX$="CREDIT    MANPOWER  EFFICIEN
CY* VEINS * "
3060 RETURN
```

Sound routines. Common to parts one and two.

```
10000 DATA T,162,400,I,108,400,T,81,60
0,A,64,100
10010 DATA 8,8,8,8,8,20,28,20,28,20,62
,54,99
10020 DATA 240,208,240,97,255,241,144,
216
20000 FOR UZ=15 TO 0 STEP -0.5:FOR UY=
3 TO 0 STEP -1:SOUND 0,15-UY,10,UZ:NEX
T UY:NEXT UZ:RETURN
20010 FOR UZ=16 TO 0 STEP -2:FOR UY=0
TO 5:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
UZ:RETURN
20020 FOR UZ=10 TO 0 STEP -2:FOR UY=5
TO 10:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
 UZ:RETURN
20030 FOR UZ=2 TO 0 STEP -2:FOR UY=50
TO 100 STEP 10:SOUND 0,150-UY,10,UZ:NE
XT UY:NEXT UZ:RETURN
20040 SOUND 0,UN,10,2:FOR UZ=1 TO 2:NE
XT UZ:SOUND 0,0,0,0:RETURN
20050 SOUND 0,255,10,4:FOR UZ=1 TO 9:N
EXT UZ:SOUND 0,0,0,0:RETURN
20060 FOR UZ=16 TO 0 STEP -1:FOR UY=15
0 TO 200 STEP 10:SOUND 0,UY,10,UZ:NEXT
 UY:NEXT UZ:RETURN
```

```
20070 FOR UZ=8 TO 0 STEP -1:FOR UY=55
TO 105 STEP 10:SOUND 0,UY,10,UZ:SOUND
1,UY-50,90,UZ:NEXT UY:NEXT UZ:RETURN
```

PART II

```
300 GOSUB 7000:GOSUB 8000:FOR P=1 TO P
L:POKE 77,0:GOSUB 1000:NEXT P:GOSUB 60
00:GOTO 300
```

Equipment Management.

```
1000 GRAPHICS 1:SETCOLOR 2,8,4:SETCOLO
R 4,P+4,2:CH=1:AC=0:C2=0
1010 POSITION 3,0:? #6;"BASE:";A$(P*10
-9,P*10):POSITION 1,2:? #6;DA;"/1/";YE
;" STATUS ";K(P,0):GOSUB J2
1020 POSITION 0,4:? #6;"EQUIPMENT OWNE
D COST":POSITION 0,6:FOR Z=1 TO 6:? #6
;EQ$(Z*10-9,Z*10):NEXT Z
1030 FOR Z=1 TO 6:POSITION 12,5+Z:? #6
;K(P,Z);" ":POSITION 16,5+Z:? #6;C(Z):
NEXT Z
1040 POSITION 7,15:? #6;"credit":POSIT
ION 8,17:? #6;K(P,13);"  ":GOSUB 1100:
IF CH=7 THEN CH=1:AC=0:GOTO 1300
1050 GOTO 1020
```

Buy and sell routine. Accessed by Equipment Management and Labor Relations phases.

```
1100 POKE 752,1:POKE 77,0:? "}";:IF CH
>7 THEN CH=CH-7
1110 IF AC=0 THEN ? ,"   INCREASE":UN=
60:GOSUB J4
1120 IF AC THEN ? ,"   DECREASE":UN=24
3:GOSUB J4
1125 IF C2=0 THEN ? :? ,"   ";EQ$(CH*1
0-9,CH*10):GOTO 1135
1130 ? :? ,"   ";WC$(CH*10-9,CH*10)
1135 IF CH=7 THEN ? :UN=121:GOSUB J4
1140 IF STICK(0)=15 AND STRIG(0)<>0 TH
EN 1140
1150 IF STRIG(0)=0 THEN 1200
1160 IF STICK(0)=11 THEN AC=0
1170 IF STICK(0)=7 THEN AC=1
1180 IF STICK(0)=13 THEN CH=CH+1
1190 IF STICK(0)=14 THEN CH=CH-1:IF CH
<1 THEN CH=7
1195 GOTO 1100
1200 IF CH=7 THEN RETURN
1205 IF C2 THEN CH=CH+6
1210 IF (AC=1 AND CH=6 AND K(P,6)=1) O
R (AC=0 AND CH=8 AND K(P,8)>17) OR (AC
=1 AND K(P,CH)<1) THEN 1280
1220 IF AC=0 AND CH=9 AND K(P,9)=0 THE
N 1250
1230 IF AC=0 AND K(P,13)<C(CH) THEN 12
80
1240 IF AC THEN K(P,CH)=K(P,CH)-1:K(P,
13)=K(P,13)+C(CH):IF CH=7 THEN K(P,14)
=K(P,14)-1
1250 IF AC=0 THEN K(P,CH)=K(P,CH)+1:K(
P,13)=K(P,13)-C(CH):IF CH=7 THEN K(P,1
4)=K(P,14)+1
1260 SETCOLOR 2,11,0:IF C2 THEN CH=CH+
1:IF CH=8 THEN SETCOLOR 2,8,4:RETURN
1270 GOSUB J3:SETCOLOR 2,8,4:RETURN
1280 SETCOLOR 2,3,0:GOSUB J5:SETCOLOR
2,8,4:IF C2 THEN CH=CH+1
1290 RETURN
```

Labor Relations phase.

```
1300 GRAPHICS 1:SETCOLOR 2,8,4:SETCOLO
R 4,P+3,2:C2=1
1310 POSITION 3,0:? #6;"BASE:";A$(P*10
-9,P*10):POSITION 4,2:? #6;"MANPOWER "
;K(P,14):POSITION 0,4:GOSUB J2
1320 K(P,7)=K(P,14)-(K(P,1)*LV)-(K(P,2
)*LV*2)-(K(P,3)*LV*3)-(K(P,4)*LV*4)-(K
(P,5)*LV*5)-(K(P,6)*LV*10)
1330 C(9)=K(P,14):C(10)=K(P,14)*2:C(11
)=K(P,14)*3:C(12)=K(P,14)*4:EN=K(P,6)*
500*(11-LV)
1340 GOSUB 5000:? #6;"EFFICIENCY RATE
";K(P,15);"%"
1350 POSITION 0,6:? #6;"conditions  se
t cost":POSITION 0,8:FOR Z=1 TO 6:? #6
;WC$(Z*10-9,Z*10):NEXT Z
1360 FOR Z=1 TO 6:POSITION 12,7+Z:? #6
;K(P,Z+6);" ":POSITION 16,7+Z:? #6;C(Z
+6):NEXT Z
1370 POSITION 7,15:? #6;"credit":POSIT
ION 8,17:? #6;K(P,13);"   ":GOSUB 1100
:IF CH=7 THEN 2000
```

```
1380 GOSUB 5000:POSITION 16,4:? #6;EF;
"% ";:GOTO 1360
```

Mining phase and graphics display.

```
2000 IF K(P,1)<1 THEN 3000
2010 GRAPHICS 7:SETCOLOR 0,1,6:SETCOLO
R 2,0,0:SETCOLOR 4,0,0:COLOR 3:PLOT 0,
19:DRAWTO 159,19
2020 COLOR 1:FOR Z=20 TO 79:SOUND 0,20
0-Z*2,10,2:PLOT 0,Z:DRAWTO 159,Z:NEXT
Z:SETCOLOR 2,11,2:GOSUB J0
2030 COLOR 1:FOR Z=1 TO 10:FOR Y=1 TO
2:R=INT(RND(0)*159):PLOT R,Z:NEXT Y:NE
XT Z
```

Set the location of the Dilithium 3 crystals.

```
2040 V1=INT(RND(0)*129+15):V2=INT(RND(
0)*59+20):RA=10-LV:RB=INT(RND(0)*3)
```

Set up the missile graphics.

```
2100 POKE 559,62:Z=PEEK(106)-8:POKE 54
279,Z:POKE 53277,3:X=Z*256+1280:RESTOR
E :FOR Z=1 TO 12:READ Z$:NEXT Z:HO=0
```

Read the data for the robominer and drill rig into the missile graphics screen display locations.

```
2110 FOR Z=57 TO 69:READ Y:POKE X+Z,Y:
NEXT Z:X=X+256:FOR Z=62 TO 69:READ Y:P
OKE X+Z,Y:NEXT Z
```

Set the playfield so that players have priority.

```
2120 SETCOLOR 2,11,2:POKE 623,1:POKE 7
52,1:IF K(P,1)=0 AND (HO=0 OR K(P,2)=0
) THEN 3000
2130 ? "}":? ,,"DRILL RIG(S) ";:X=1:W=
200:GOTO 2150
2140 ? "}":? "ROBOMINER(S) ";:X=2:W=50
2150 ? K(P,X):POKE 53248+X,W:Z=704+X:P
OKE Z,78:GOSUB J0+X*10
2160 FOR Y=78 TO 64 STEP -1:POKE Z,Y:I
F STRIG(0)=0 THEN Y=64:NEXT Y:POKE Z,7
8:GOTO 2200
2165 IF STICK(0)=7 THEN Y=0:NEXT Y:POK
E 53250,0:GOTO 2130
2170 IF STICK(0)=11 THEN Y=0:NEXT Y:PO
KE 53249,0:GOTO 2140
2175 IF STICK(0)=14 THEN 2185
2180 NEXT Y:GOTO 2160
2185 ? "}":? ," CONTINUE";:POKE 5324
9,0:POKE 53250,0:GOSUB J3:IF STRIG(0)=
0 THEN 3000
2190 FOR Y=78 TO 64 STEP -1:IF STICK(0
)=15 THEN 2185
2195 GOTO 2165
2200 IF K(P,X)<1 OR (X=2 AND HO=0) THE
N SETCOLOR 2,3,2:GOSUB J5:SETCOLOR 2,1
1,2:GOTO 2160
2210 K(P,X)=K(P,X)-1:FOR Z=20 TO 0 STE
P -1:SOUND 0,Z,10,2:NEXT Z
```

Horizontal movement phase.

```
2220 SOUND 0,W,10,2:IF STRIG(0)=0 THEN
 2300
2230 IF STICK(0)=11 THEN W=W-1:IF W<50
 AND X=1 THEN W=50
2235 IF W<40 THEN W=40
2240 IF STICK(0)=7 THEN W=W+1:IF W>200
 THEN W=200
2250 POKE 53248+X,W:GOTO 2220
2300 IF X=1 AND (W=50 OR W=200) THEN 2
220
2310 IF X=2 THEN 2400
2320 W=W-44:Y=19
```

Vertical drilling sequence.

```
2330 COLOR 2:FOR Z=10 TO 14:SETCOLOR 1
,0,Z:PLOT W,19:VE=1
2333 IF STICK(0)=13 THEN Y=Y+1:SOUND 0
,200,2,8:VE=2:IF Y>78 THEN Y=78
2335 GOSUB 2500:? ,"ENERGY ";EN:IF EN<
1 OR VE=1 THEN Z=14:NEXT Z:GOTO 2600
2340 SOUND 0,150,12,2:DRAWTO W,Y:NEXT
Z:IF STRIG(0)<>0 THEN 2330
2350 COLOR 1:FOR Z=16 TO 18:PLOT W-4,Z
:DRAWTO W+4,Z:NEXT Z:COLOR 0:PLOT W,Y:
DRAWTO W,17:POKE 53249,0:GOSUB J1
2360 HO=HO+1:GOTO 2120
```

Robominer vertical drilling sequence.

```
2400 LOCATE W-41,19,Z:IF Z<>0 THEN 222
0
2410 W=W-41:Y=17
```

```
2420 COLOR 2:FOR Z=10 TO 14:SETCOLOR 1
,0,Z:PLOT W,17:IF STICK(0)=13 THEN Y=Y
+1:LOCATE W,Y,X:IF X<>0 THEN Y=Y-1
2430 IF STICK(0)=14 THEN COLOR 0:PLOT
W,Y:Y=Y-1:IF Y<20 THEN Y=20
2435 VE=1:GOSUB 2500:SOUND 0,150,12,2:
DRAWTO W,Y:? ,"ENERGY ";EN:IF EN<1 THE
N Z=14:NEXT Z:GOTO 2600
2440 NEXT Z:IF (STICK(0)<>7 AND STICK(
0)<>11) OR Y<20 THEN 2420
2445 COLOR 0:PLOT W,17:DRAWTO W,Y
```

Robominer horizontal drilling sequence.

```
2450 VE=1:COLOR 0:SETCOLOR 1,0,15:PLOT
 W,Y:IF STICK(0)=7 THEN W=W+1:SOUND 0,
200,2,8:VE=2:IF W>150 THEN W=W-1
2460 IF STICK(0)=11 THEN W=W-1:SOUND 0
,200,2,8:VE=2:IF W<10 THEN W=W+1
2465 GOSUB 2500:? ,"ENERGY ";EN:IF EN<
1 OR VE=1 THEN 2600
2470 SOUND 0,150,12,2:COLOR 2:PLOT W,Y
:IF STRIG(0)<>0 THEN 2450
2480 POKE 53250,0:GOSUB J1:GOTO 2120
```

Compare distance of drill rig or robominer to Dilithium.

```
2500 POKE 77,0:EN=INT(EN-(100/K(P,15))
*VE)
2510 VE=0:? "}":IF Y<20 THEN SETCOLOR
2,11,2:? "SURFACE      ";:RETURN
2520 IF (W<V1-RA*3 OR W>V1+RA*4) OR (Y
<V2-RB*3 OR Y>V2+K(P,4)+RB*4) THEN SET
COLOR 2,1,2:? "ICE  AND ROCK";:RETURN
2530 IF (W<V1-RA*2 OR W>V1+RA*3) OR (Y
<V2-RB*2 OR Y>V2+K(P,4)+RB*3) THEN SET
COLOR 2,7,2:? "THIRIDIUM    ";:RETURN
2540 IF (W<V1-RA OR W>V1+RA*2) OR (Y<V
2-RB OR Y>V2+K(P,4)+RB*2) THEN SETCOLO
R 2,5,2:? "SECONIUM     ";:RETURN
2550 IF (W<V1 OR W>V1+RA) OR (Y<V2 OR
Y>V2+K(P,4)+RB) THEN SETCOLOR 2,3,2:?
"FIRIDIUM     ";:RETURN
2560 VE=1:RETURN
```

Discovery of Dilithium.

```
2600 IF EN<1 THEN 2800
2605 COLOR 3:PLOT W,Y:? "}":? "EUREKA!
 You have found DILITHIUM!!!":FOR Y=1
TO 10:FOR Z=15 TO 0 STEP -1
2610 SETCOLOR 2,3,Z:SOUND 0,160-(Z*10)
,10,Y:NEXT Z:NEXT Y:SOUND 0,0,0,0:GOSU
B J0:SETCOLOR 2,11,2
2620 ? "}":? ,,,"STATUS":GOSUB J1:X=0:
Y=1:GOTO 2640
2630 ? "}":? "500 CREDITS":GOSUB J2:X=
13:Y=500
2640 IF STRIG(0)=0 THEN K(P,X)=K(P,X)+
Y:K(P,16)=K(P,16)+1:GOTO 2700
2650 IF STICK(0)=7 THEN 2620
2660 IF STICK(0)=11 THEN 2630
2670 GOTO 2640
```

Test for lack of energy or equipment for drilling.

```
2700 IF EN<1 THEN 2800
2710 POKE 53250,0:POKE 53249,0:IF K(P,
1)>0 THEN 2000
2720 ? "}":? "YOU DO NOT HAVE ENOUGH E
QUIPMENT LEFT":? "TO CONTINUE DRILLING
.":GOSUB J6:GOTO 3000
2800 POKE 53250,0:POKE 53249,0
2810 ? "}":? "YOU DO NOT HAVE ENOUGH E
NERGY LEFT TO":? "   CONTINUE DRILLING
.":GOSUB J6:GOTO 3000
```

Cave-in sequence.

```
3000 R=RND(0)*10-K(P,11):IF R<7 OR K(P
,16)<1 THEN 3500
3010 GRAPHICS 18:SETCOLOR 4,3,0:POSITI
ON 5,0:? #6;"RED ALERT!":Z=0
3020 FOR Y=30 TO 0 STEP -1:SOUND 0,150
-Y,10,6:SETCOLOR 0,14,Y:NEXT Y
3025 Z=Z+1:IF Z<5 THEN 3020
3030 SETCOLOR 0,14,10:SOUND 0,0,0,0:PO
SITION 1,3:? #6;"CAVE IN:":R=INT(RND(0
)*11)
3040 POSITION 3,5:? #6;R;" WORKERS LOS
T!":K(P,14)=K(P,14)-R
3050 IF R>6 THEN POSITION 3,6:? #6;"SH
AFT NOW CLOSED!":K(P,16)=K(P,16)-1
```

```
3060 IF STRIG(0) THEN 3060
3070 GOTO 4000
```

Meteor-storm sequence.

```
3500 R=RND(0)*10-K(P,3):IF R<8 THEN 40
00
3510 GRAPHICS 18:SETCOLOR 4,3,0:POSITI
ON 5,0:? #6;"RED ALERT!":Z=0
3520 FOR Y=30 TO 0 STEP -1:SOUND 0,150
-Y,10,6:SETCOLOR 0,14,Y:NEXT Y:Z=Z+1:I
F Z<5 THEN 3520
3530 SETCOLOR 0,14,10:SOUND 0,0,0,0:PO
SITION 1,3:? #6;"METEOR STORM!:":R=INT
(RND(0)*10+1)
3540 POSITION 3,5:? #6;R;" WORKERS LOS
T!":K(P,14)=K(P,14)-R:IF R<7 THEN 3590
3550 POSITION 0,6:R=INT(RND(0)*5+1):IF
 K(P,R)<1 THEN 3590
3560 K(P,R)=K(P,R)-1:? #6;EQ$(10*R-9,1
0*R);"DESTROYED!"
3590 IF STRIG(0) THEN 3590
```

Monthly report.

```
4000 Z=K(P,5):IF K(P,16)<Z THEN Z=K(P,
16)
4010 K(P,13)=INT(K(P,13)+K(P,13)*0.1+5
00+(50*K(P,0))+(Z*25*K(P,8)))
4020 GRAPHICS 0:SETCOLOR 2,11,0:SETCOL
OR 4,11,0:POKE 752,1:GOTO 4100
4030 FOR Z=1 TO 36:? Z$;:NEXT Z:RETURN
4100 ? CHR$(2);:Z$=CHR$(13):GOSUB 4030
:? CHR$(22);
4105 ? CHR$(2);:POSITION 19,1:? "TITAN
":POSITION 39,1:? CHR$(22);" ";CHR$(2)
;:POSITION 39,2:? CHR$(22);
4110 ? CHR$(2);:? " MONTHLY REPORT":PO
SITION 29,3:? DA;"/1/";YE:POSITION 39,
3:? CHR$(22);" ";CHR$(2);:POSITION 39,
4:? CHR$(22)
4120 ? CHR$(2);:Z$=CHR$(18):GOSUB 4030
:? CHR$(22);:GOSUB J7
4130 ? "! ITEM       /ACTA /BELL /CHIM
 /DAED /";
4131 REM Line 4130: Replace ! with
<CTRL>B. Replace / with <CTRL>V
4140 ? CHR$(2);:Z$=CHR$(18):GOSUB 4030
:? CHR$(22);
4144 REM Lines 4145-4195: Replace !
with <CTRL>B. Replace / with <CTRL>V
4145 ? "!              /     /     /
 /     /";
4150 Y=8:FOR Z=0 TO 6:Y=Y+1:POSITION 1
,Y:W=Z*10:IF W=0 THEN ? " ! STATUS":GO
TO 4170
4160 ? " ! ";EQ$(W-9,W)
4170 FOR X=1 TO 4:POSITION 6*X+9,Y:? "
/";K(X,Z):POSITION 39,Y:? "/";:NEXT X:
NEXT Z
4180 FOR Z=1 TO 4:Y=Y+1:POSITION 1,Y:?
 " ! ";EX$(Z*10-9,Z*10)
4190 FOR X=1 TO 4:POSITION 6*X+9,Y:? "
/";K(X,Z+12):POSITION 39,Y:? "/";:NEXT
 X:NEXT Z
4195 ? "!              /     /     /
 /     /";
4200 ? " ";:FOR Z=1 TO 36:? CHR$(18);:
NEXT Z
4210 POSITION 8,23:? "Press TRIGGER to
 continue.";:GOSUB J6
4220 IF STRIG(0)=0 THEN GOSUB J5:RETUR
N
4230 GOTO 4220
```

Efficiency rating algorithms.

```
5000 FF=0:FOR Z=1 TO 5:FF=FF+K(P,Z)*Z:
NEXT Z:IF FF<1 THEN FF=1
5010 EF=K(P,6)*10/FF:IF EF>1 THEN EF=F
F/(K(P,6)*10)
5020 EF=EF*100:FF=K(P,7)*2.5:IF FF<0 T
HEN FF=FF*-2
5030 EF=EF-FF:FF=(K(P,8)-6)*3:IF FF<0
THEN FF=FF*-2
5040 EF=EF-FF+(K(P,9)*5)+(K(P,10)*5)+(
K(P,11)*7.5)+(K(P,12)*10):EF=INT(EF):I
F EF<5 THEN EF=5
5050 IF EF>100 THEN EF=100
5060 IF K(P,9)=0 THEN EF=1
5070 K(P,15)=EF:RETURN
```

New cost algorithms.

```
6000 FOR Z=1 TO 8:X=0:FOR Y=1 TO 4:X=X
+K(Y,Z):NEXT Y:X=X/PL
6010 IF Z>5 THEN C(Z)=INT(C(Z)+(C(Z)*X
/(55-LV))):GOTO 6050
6020 C(Z)=INT(C(Z)+(C(Z)*X/(25-LV))-(C
(Z)*1/(25-LV)))
6030 IF C(Z)<100*Z THEN C(Z)=100*Z
6040 IF C(Z)>200*Z THEN C(Z)=200*Z
6050 IF Z>6 AND C(Z)<1 THEN C(Z)=1
6060 IF Z>6 AND C(Z)>50 THEN C(Z)=50
6070 IF Z=6 AND C(6)>4000 THEN C(Z)=40
00
6080 NEXT Z:FOR Z=1 TO 4:FOR Y=8 TO 12
:K(Z,Y)=0:NEXT Y:NEXT Z:RETURN
```

Adjust date.

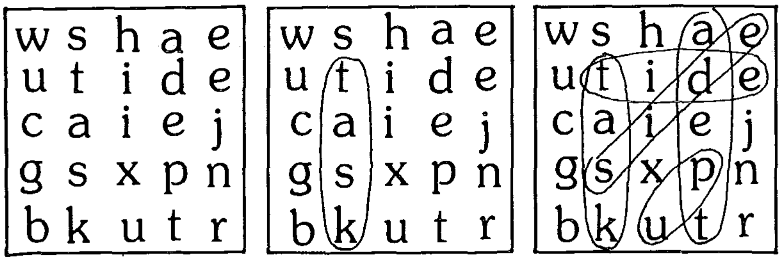
```
7000 DA=DA+1:IF DA=13 THEN DA=1:YE=YE+
1
7010 RETURN
```

End of game.

```
8000 IF YE=2051 AND DA>3 THEN 8500
8010 R=INT(RND(0)*2):IF R=0 OR YE=2050
 THEN RETURN
8500 POSITION 6,23:? "The INSPECTORS h
ave arrived!";:FOR Y=1 TO 10:FOR Z=15
TO 0 STEP -1
8510 SETCOLOR 2,11,Z:SOUND 0,160-(Z*10
),10,Y:NEXT Z:NEXT Y:SOUND 0,0,0,0:GOS
UB J0
8520 POSITION 3,23:? "Your performance
 is being assessed.";:FOR Z=1 TO 200:N
EXT Z:IF PL=1 THEN 8600
8530 Z=-1:Y=0:FOR X=1 TO PL:IF K(X,0)>
=Z THEN Z=K(X,0):Y=Z
8540 NEXT X:IF Y=1 THEN 8590
8550 FOR X=1 TO Y-1:IF K(X,0)=Z THEN 8
570
8560 NEXT X:GOTO 8590
8570 FOR X=1 TO Y:IF K(X,0)<0 THEN K(X
,0)=0:NEXT X:GOTO 8530
8580 K(X,0)=K(X,16)*100+K(X,15)+K(X,14
)+K(X,13)+RND(1):NEXT X:GOTO 8530
8590 SETCOLOR 4,5,0:SETCOLOR 2,5,0:POS
ITION 4,22:? "The supervisor of";A$(Y*
10-9,Y*10);"has won":GOTO 8640
8600 IF K(1,0)>19 THEN 8630
8610 SETCOLOR 4,3,0:SETCOLOR 2,3,0:POS
ITION 2,22:? "I am sorry to report tha
t you did not";
8620 POSITION 0,23:? "achieve enough s
tatus to gain all TITAN";:GOSUB 20060:
GOTO 8700
8630 SETCOLOR 4,8,0:SETCOLOR 2,8,0:POS
ITION 4,22:? "Because of your status y
ou have won"
8640 ? "   the right to mine all of TI
TAN!!  ";:GOSUB J7
8700 GOTO 8700
```

Sound routines. Common to parts one and two.

```
10000 DATA T,162,400,I,108,400,T,81,60
0,A,64,100
10010 DATA 8,8,8,8,8,20,28,20,28,20,62
,54,99
10020 DATA 240,208,240,97,255,241,144,
216
20000 FOR UZ=15 TO 0 STEP -0.5:FOR UY=
3 TO 0 STEP -1:SOUND 0,15-UY,10,UZ:NEX
T UY:NEXT UZ:RETURN
20010 FOR UZ=16 TO 0 STEP -2:FOR UY=0
TO 5:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
UZ:RETURN
20020 FOR UZ=10 TO 0 STEP -2:FOR UY=5
TO 10:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
 UZ:RETURN
20030 FOR UZ=2 TO 0 STEP -2:FOR UY=50
TO 100 STEP 10:SOUND 0,150-UY,10,UZ:NE
XT UY:NEXT UZ:RETURN
20040 SOUND 0,UN,10,2:FOR UZ=1 TO 2:NE
XT UZ:SOUND 0,0,0,0:RETURN
20050 SOUND 0,255,10,4:FOR UZ=1 TO 9:N
EXT UZ:SOUND 0,0,0,0:RETURN
20060 FOR UZ=16 TO 0 STEP -1:FOR UY=15
0 TO 200 STEP 10:SOUND 0,UY,10,UZ:NEXT
 UY:NEXT UZ:RETURN
20070 FOR UZ=8 TO 0 STEP -1:FOR UY=55
TO 105 STEP 10:SOUND 0,UY,10,UZ:SOUND
1,UY-50,90,UZ:NEXT UY:NEXT UZ:RETURN
```



Word Search Puzzle Generator

by David W. Durkee **Atari version by Jon R. Voskuil**

***Word-Search Puzzle Generator* is for an Atari with at least 8K RAM and a printer.**

Word-Search Puzzle Generator is a clever program that lets you create entertaining puzzles in little more time than it takes to imagine the words for them.

Upon running the program, you will receive the option of seeing the puzzle as it is created or leaving the screen blank so that you can enjoy working the puzzle yourself later. Next, you simply type words to your heart's content. The Atari does the rest, placing your words in random orientations in the 37-by-20-character letter matrix.

The size of the screen display was the reason for the choice of these dimensions. If you are interested only in the print-out of your puzzle, you may easily alter the size of the matrix by changing the DIMension statement and the various loops that use those dimensions.

After you have typed all the words you wish to include in your puzzle, type the word *STOP.* This will instruct the computer to print an answer key, the complete puzzle with random letters filled in, and a list of the words you entered. After this, you may elect to print additional copies of the puzzle.

Besides the obvious entertainment value of word-search puzzles, there is also great potential for educational applications. Working this type of puzzle is an easy way to become familiar with a list of words, whatever the subject matter might be.

Variables

A(i,j): Array that stores the ASCII codes for letters in the letter matrix.
A$: Input variable. Also used to assemble each line of the puzzle as it is printed.
B: Counts the directions in which a word may go.
B(i,j): Notes the directions in which the computer may draw a word, given a random starting position. If B(X+2,Y+2) is 1, then the word shares at least one letter with other words.
C: Loop counter.
D, R: Used to select the best direction in the B matrix.
L, U: Random starting co-ordinates for a word.
X, Y: Indicate word direction along the x and y axes; values can be -1, 0, or 1 (but not both may be 0), defining the eight diferent directions.
X1, Y1: Printing co-ordinates for individual letters; derived from U, L, X, and Y.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS   'Word Search Puzzle'     SS
SS    Author: David Durkee    SS
SS Translator: Jon R. Voskuil SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

Initialization and instructions.

```
90 POKE 752,1
100 ? CHR$(125):POSITION 10,5:PRINT "W
ORD SEARCH PUZZLE"
105 POSITION 10,8:PRINT "BY DAVID W. D
URKEE":POSITION 10,10:PRINT "COPYRIGHT
 (C) 1981"
107 POSITION 7,13:PRINT "TRANSLATED BY
 JON VOSKUIL"
110 FOR I=1 TO 2000:NEXT I
120 ? CHR$(125):POSITION 2,5:PRINT "TO
 CREATE A PUZZLE, SIMPLY ENTER A    WO
RD WHICH YOU WOULD LIKE TO HAVE IN"
125 PRINT "THE PUZZLE AFTER THE '?' PR
OMPT."
130 PRINT :PRINT "WHEN YOU'VE ENTERED
ALL THE WORDS YOU WOULD LIKE IN THE PU
ZZLE, ENTER THE"
135 PRINT "WORD 'STOP' AND THE ATARI W
ILL DO THE REST."
140 PRINT :PRINT "IF YOU WOULD LIKE A
PUZZLE FOR YOUR-  SELF (BLANK SCREEN),
 THEN TYPE '1';"
145 PRINT "OTHERWISE, TYPE '0' TO BEGI
N: ";
150 INPUT BLANK:PRINT "}":Z=0
160 DIM W$(1000),A$(41),B(3,3),A(37,20
)
165 W$="":FOR I=1 TO 37:FOR J=1 TO 20:
A(I,J)=0:NEXT J:NEXT I
```

Beginning of word-entry loop.

```
170 Z=Z+1
180 POSITION 2,22:PRINT "WORD #";Z;":
";:INPUT A$:IF A$="" THEN 180
190 IF A$="STOP" THEN 530
```

Choose random starting location.

```
200 W$(LEN(W$)+1)=A$:W$(LEN(W$)+1)="*"
210 U=INT(RND(1)*20)+1:L=INT(RND(1)*37
)+1
```

Check each direction to see if word can be written that way.

```
220 FOR X=-1 TO 1:FOR Y=-1 TO 1
230 IF X=Y AND Y=0 THEN 330
240 X1=L:Y1=U
250 FOR C=1 TO LEN(A$)
260 X1=X1+X:Y1=Y1+Y
270 IF X1>37 OR X1<1 OR Y1>20 OR Y1<1
THEN B(X+2,Y+2)=0:GOTO 330
280 IF A(X1,Y1)=0 THEN 310
290 IF A(X1,Y1)<>ASC(A$(C,C)) THEN B(X
+2,Y+2)=0:GOTO 330
300 B(X+2,Y+2)=B(X+2,Y+2)+1
310 NEXT C
320 B(X+2,Y+2)=B(X+2,Y+2)+1:B=B+1
330 NEXT Y:NEXT X
340 IF B=0 THEN 210
```

Select direction. If possible, make word intersect with another.

```
350 R=2:D=2
360 FOR X=1 TO 3:FOR Y=1 TO 3
370 IF B(X,Y)>B(R,D) THEN R=X:D=Y
380 NEXT Y:NEXT X
390 X=R-2:Y=D-2
400 IF X=-1 AND Y=-1 AND B(1,1)=1 THEN
 420
410 GOTO 440
420 X=INT(RND(1)*3)-1:Y=INT(RND(1)*3)-
1
430 IF (X=0 AND Y=0) OR B(X+2,Y+2)=0 T
HEN 420
440 X1=L:Y1=U
```

Print word on screen, unless user chose a blank screen.

```
450 FOR C=1 TO LEN(A$)
460 X1=X1+X:Y1=Y1+Y
470 A(X1,Y1)=ASC(A$(C,C))
480 IF BLANK THEN 500
490 POSITION X1+1,Y1:PRINT CHR$(A(X1,Y
1));
500 NEXT C
510 B=0:FOR X=1 TO 3:FOR Y=1 TO 3:B(X,
Y)=0:NEXT Y:NEXT X
520 POSITION 2,22:PRINT "
                         ";:GOTO 170
```

Prepare answer key.

```
530 FOR X=1 TO 37:FOR Y=1 TO 20
540 IF A(X,Y)<>0 THEN 560
550 A(X,Y)=45:POSITION X+1,Y:PRINT "-"
;
560 NEXT Y:NEXT X
570 POSITION 2,22:PRINT "READY TO PRIN
T; TURN ON PRINTER AND   HIT 'RETURN'
";:INPUT A$:GOSUB 680
580 LPRINT :LPRINT "WORD PUZZLE ANSWER
 KEY"
590 LPRINT :LPRINT :LPRINT
```

Fill in blanks with random letters.

```
600 PRINT :PRINT "PLEASE WAIT A MOMENT
 FOR ME TO CREATE PUZZLE. . ."
610 FOR X=1 TO 37:FOR Y=1 TO 20
620 IF A(X,Y)<>45 THEN 640
630 B=INT(RND(1)*26)+65:A(X,Y)=B
640 NEXT Y:NEXT X
650 GOSUB 680
660 LPRINT :LPRINT "COMPUTER GENERATED
 WORD PUZZLE"
670 LPRINT :LPRINT :LPRINT :GOTO 730
```

Print complete puzzle.

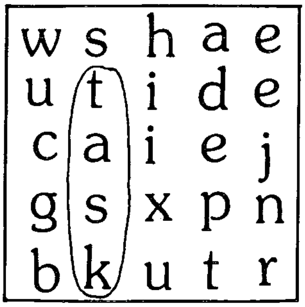
```
680 LPRINT
690 FOR X=1 TO 37:A$=""
692 FOR Y=1 TO 20
695 A$(LEN(A$)+1)=CHR$(A(X,Y))
697 A$(LEN(A$)+1)=" "
700 NEXT Y
705 A$=A$(1,LEN(A$)-1)
710 LPRINT A$:NEXT X
720 RETURN
```

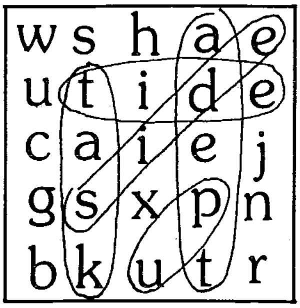
Print word list.

```
730 LPRINT :LPRINT :LPRINT "WORD LIST"
:LPRINT
740 J=1:FOR I=1 TO LEN(W$)
744 IF W$(I,I)="*" THEN LPRINT W$(J,I-
1):J=I+1
746 NEXT I
```

Another copy? If not, then end.

```
760 PRINT :PRINT "WOULD YOU LIKE ANOTH
ER COPY ";:INPUT A$
765 IF A$(1,1)="Y" THEN LPRINT :LPRINT
 :LPRINT :LPRINT :GOTO 650
770 END
```





FUGUE

by William Morris and John Cope

***Fugue* is a musical program for an Atari with 16K RAM.**

For lovers of serious music, here's a program that will play a Bach fugue for you, and let you watch it in color on your video screen as well. While this program does not permit you to control the actual music, as many software packages do, it does have four voices and an intricate interplay thereof, uncommon for the Atari.

Type it in and relax. We think you'll enjoy it.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS          'Fugue'           SS
SS          Authors:          SS
SS William Morris & John Cope SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
1 REM (c) 1980 Wm. Morris & J. Cope
10 DIM W(3),X(3),Y(3):R=RND(0)*15
```

Title page.

```
20 GRAPHICS 2+16:SETCOLOR 4,3,2:COLOR
32:PLOT 6,5:? #6;"FUGUE":FOR TT=1 TO 2
000:NEXT TT
```

Draw four lines, and place the "notes" at the left edge.

```
210 GRAPHICS 3+16:SETCOLOR 0,8,9:SETCO
LOR 1,8,5:SETCOLOR 2,9,2:SETCOLOR 4,R,
0
220 COLOR 3
230 FOR Z=0 TO 3:PLOT 0,Z*5+3:DRAWTO 3
9,Z*5+3:NEXT Z
240 FOR Z=0 TO 3:W(Z)=0:X(Z)=3:NEXT Z
```

Jump to line 330 when an OUT OF DATA error occurs.

```
250 TRAP 330
```

For each of the 4 voices (Z = Ø to 3), read the note (Y), then plot it at the proper position (W) on the proper line (X).

```
300 FOR Z=0 TO 3:READ Y:Y(Z)=Y:COLOR 3
:PLOT W(Z),X(Z):COLOR 1:W(Z)=INT(Y(Z)/
6):X(Z)=Z*5+3:PLOT W(Z),X(Z):NEXT Z
310 SOUND 0,Y(0),10,4:SOUND 1,Y(1),10,
4:SOUND 2,Y(2),10,4:SOUND 3,Y(3),10,4:
GOTO 300
```

Turn off the 4 voices, then restart the program.

```
330 FOR Z=0 TO 3:SOUND Z,0,0,0:NEXT Z:
```

```
FOR Z=1 TO 500:NEXT Z:RUN
```

Data.

```
1000 DATA 81,0,0,0,81,0,0,0,81,0,0,0,8
1,0,0,0
1010 DATA 53,0,0,0,53,0,0,0,53,0,0,0,5
3,0,0,0
1020 DATA 68,0,0,0,68,0,0,0,68,0,0,0,6
8,0,0,0
1030 DATA 68,0,0,0,68,0,0,0,72,0,0,0,7
2,0,0,0
1040 DATA 81,0,0,0,81,0,0,0,68,0,0,0,6
8,0,0,0
1050 DATA 72,0,0,0,72,0,0,0,81,0,0,0,8
1,0,0,0
1060 DATA 85,0,0,0,85,0,0,0,72,0,0,0,7
2,0,0,0
1070 DATA 108,0,0,0,108,0,0,0,108,0,0,
0,108,0,0,0
1080 DATA 81,0,0,0,81,0,0,0,108,0,0,0,
108,0,0,0
1090 DATA 72,0,0,0,72,0,0,0,108,0,0,0,
108,0,0,0
1100 DATA 68,0,0,0,68,0,0,0,72,0,0,0,8
1,0,0,0
1110 DATA 72,0,0,0,72,0,0,0,108,0,0,0,
108,0,0,0
1120 DATA 81,0,0,0,81,0,0,0,108,0,0,0,
81,0,0,0
1130 DATA 72,0,0,0,72,0,0,0,108,0,0,0,
72,0,0,0
1140 DATA 68,0,0,0,68,0,0,0,72,0,0,0,8
1,0,0,0
1150 DATA 72,0,0,0,108,0,0,0,53,0,0,0,
60,0,0,0
1160 DATA 68,0,0,0,72,0,0,0,81,0,0,0,6
8,0,0,0
1170 DATA 72,0,0,0,81,0,0,0,85,0,0,0,7
2,0,0,0
1180 DATA 81,0,0,0,108,0,0,0,81,0,0,0,
72,0,0,0
1190 DATA 68,0,0,0,60,0,0,0,53,0,0,0,4
7,0,0,0
1200 DATA 45,108,0,0,47,108,0,0,53,108
,0,0,45,108,0,0
1210 DATA 47,72,0,0,53,72,0,0,57,72,0,
0,47,72,0,0
1220 DATA 53,91,0,0,53,91,0,0,72,91,0,
0,72,91,0,0
1230 DATA 53,91,0,0,53,91,0,0,47,96,0,
0,47,96,0,0
1240 DATA 45,108,0,0,40,108,0,0,45,91,
0,0,40,91,0,0
1250 DATA 45,96,0,0,40,96,0,0,45,108,0
,0,40,108,0,0
1260 DATA 35,114,0,0,40,114,0,0,35,96,
0,0,33,96,0,0
1270 DATA 35,144,0,0,40,144,0,0,45,144
,0,0,47,144,0,0
1280 DATA 45,108,0,0,35,108,0,0,40,144
,0,0,35,144,0,0
1290 DATA 57,96,0,0,35,96,0,0,40,144,0
,0,35,144,0,0
1300 DATA 53,91,0,0,35,91,0,0,40,96,0,
0,35,108,0,0
1310 DATA 57,96,0,0,35,96,0,0,40,144,0
,0,35,144,0,0
1320 DATA 45,108,0,0,53,108,0,0,57,144
,0,0,53,108,0,0
1330 DATA 40,96,0,0,53,96,0,0,57,144,0
,0,53,96,0,0
1340 DATA 35,91,0,0,53,91,0,0,57,96,0,
0,53,108,0,0
1350 DATA 40,96,0,0,53,144,0,0,57,72,0
,0,53,81,0,0
1360 DATA 72,91,0,0,72,96,0,0,45,108,0
,0,45,91,0,0
1370 DATA 81,96,0,0,81,108,0,0,47,114,
0,0,47,96,0,0
1380 DATA 91,108,0,0,91,144,0,0,72,108
,0,0,72,96,0,0
1390 DATA 53,91,0,0,53,81,0,0,45,72,0,
0,45,64,0,0
1400 DATA 50,60,0,0,50,68,0,0,35,60,0,
0,35,53,0,0
1410 DATA 00,60,0,0,00,68,0,0,00,72,0,
0,50,60,0,0
1420 DATA 53,68,0,0,53,72,0,0,40,53,0,
0,40,60,0,0
```

```
1430 DATA 00,68,0,0,00,72,0,0,00,81,0,
0,53,68,0,0
1440 DATA 60,72,0,0,68,72,0,0,60,81,0,
0,53,81,0,0
1450 DATA 60,85,0,0,35,85,0,0,40,108,0
,0,35,108,0,0
1460 DATA 68,81,162,0,40,81,162,0,42,8
1,162,0,40,81,162,0
1470 DATA 72,0,108,0,45,0,108,0,47,0,1
08,0,45,0,108,0
1480 DATA 40,0,136,0,40,0,136,0,40,108
,136,0,40,108,136,0
1490 DATA 40,81,136,0,40,81,136,0,40,7
2,144,0,40,72,144,0
1500 DATA 0,68,162,0,0,60,162,0,0,68,1
36,0,0,60,136,0
1510 DATA 0,68,144,0,0,60,144,0,0,68,1
62,0,0,60,162,0
1520 DATA 0,53,173,0,0,60,173,0,0,53,1
44,0,0,50,144,0
1530 DATA 0,53,217,0,0,60,217,0,0,68,2
17,0,0,72,217,0
1540 DATA 68,108,162,0,53,108,162,0,60
,0,217,0,53,0,217,0
1550 DATA 85,121,144,0,53,121,144,0,60
,0,217,0,53,0,217,0
1560 DATA 81,108,136,0,53,108,136,0,60
,0,144,0,53,0,162,0
1570 DATA 85,121,144,0,53,121,144,0,60
,0,217,0,53,0,217,0
1580 DATA 68,00,162,0,68,81,162,0,68,8
5,217,0,68,81,162,0
1590 DATA 60,0,144,0,60,81,144,0,60,85
,217,0,60,81,162,0
1600 DATA 53,0,136,0,53,81,136,0,53,85
,144,0,53,81,162,0
1610 DATA 60,0,144,0,60,81,217,0,60,85
,108,0,60,81,121,0
1620 DATA 0,108,136,0,0,108,144,0,68,1
08,162,0,68,108,136,0
1630 DATA 0,121,144,0,0,121,162,0,72,1
21,173,0,72,121,144,0
1640 DATA 0,136,162,0,0,136,217,0,108,
136,162,0,108,136,144,0
1650 DATA 81,0,136,0,72,0,121,0,68,0,1
08,0,81,0,96,0
1660 DATA 72,0,91,217,72,0,96,217,53,0
,108,217,53,0,91,217
1670 DATA 57,0,96,144,57,0,108,144,47,
0,114,144,47,0,96,144
1680 DATA 35,0,108,182,33,0,108,182,35
,0,144,182,40,0,144,182
1690 DATA 45,0,108,182,47,0,108,182,53
,0,96,193,57,0,96,193
1700 DATA 53,0,91,217,53,0,81,217,0,0,
91,182,0,0,81,182
1710 DATA 33,0,91,193,33,0,81,193,0,0,
91,217,0,0,81,217
1720 DATA 47,0,72,230,47,0,81,230,0,0,
72,193,0,0,68,193
1730 DATA 0,0,72,230,0,0,81,230,35,0,9
1,230,35,0,96,230
1740 DATA 35,0,91,217,35,0,72,217,35,0
,81,217,35,0,72,217
1750 DATA 35,0,114,193,35,0,72,193,35,
0,81,193,35,0,72,193
1760 DATA 35,0,108,182,35,0,72,182,35,
0,81,193,35,0,72,217
1770 DATA 35,0,114,193,35,0,72,193,35,
0,81,193,35,0,72,193
1780 DATA 35,0,91,217,35,0,108,217,35,
0,114,217,35,0,108,217
1790 DATA 35,0,81,193,35,0,108,193,35,
0,114,193,35,0,108,193
1800 DATA 35,0,72,182,35,0,108,182,35,
0,114,193,35,0,108,217
1810 DATA 35,0,81,193,35,0,108,193,35,
0,114,193,35,0,108,193
1820 DATA 40,0,91,217,35,0,91,217,33,0
,96,217,35,0,108,217
1830 DATA 40,0,114,230,45,0,108,230,47
,0,114,230,53,0,108,230
1840 DATA 53,0,108,217,53,0,108,217,53
,0,108,217,53,0,108,217
1850 DATA 53,0,108,217,53,0,108,217,53
,0,108,217,53,0,108,217
1860 DATA 53,0,108,217,53,0,108,217,53
,0,108,217,53,0,108,217
1870 DATA 53,0,108,217,53,0,108,217,53
,0,108,217,53,0,108,217
```

Flight of the Bumblebee

by William Morris and John Cope

***Flight of the Bumblebee* is a musical animated graphics program for an Atari with 16K RAM.**

Those of you without culture may have to be informed that *The Flight of the Bumblebee* is not a beekeeper simulation, but rather a classic tune from Rimsky-Korsakoff's 1900 opera, *The Tale of the Tsar Saltan.* The authors of this program, whose names are familiar to regular *SoftSide* readers, have digitized and animated this great Russian composer's work to suit the music and graphic capabilities of the Atari computer.

No special instructions are needed; you just type in the program of your choice and RUN.

Variables
UN: The pitch of the note to be played by the sound routine.
UB: Memory location used in the routine to set the three graphics modes used in the title page.
UC: Memory location used in moving the redefined character set.
UZ, Z: Counters.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS 'Flight of  the Bumblebee' SS
SS           Authors:         SS
SS William Morris & John Cope SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

Set up a jump to line 19999 when the data are all read. Reserve memory for a redefined character set.

```
10 TRAP 19999:GOSUB 30300
```

Title display.

```
20 GOSUB 30200:POKE 87,2:POSITION 3,1:
? #6;"THE BUMBLE BEE"
30 POKE 87,1:POSITION 1,7:? #6;"by rim
sky korsakoff":POKE 752,1
40 POKE 87,0:POSITION 5,14:? "(c) Wm.
Morris & J. Cope  1981"
100 GOSUB 30310
```

Set up the graphics display. Line 110 establishes the colors to be used. Line 130 sets up the "text window." Line 140 draws the "grass." Lines 150-160 use redefined characters to draw the "flowers." Lines 170-190 draw the "beehive." Note: the COLOR 87, 130, 214, and 215 statements are not misprints; they are a method of printing redefined characters.

```
110 GRAPHICS 1:POKE 756,UC/255+2:SETCO
LOR 0,15,2:SETCOLOR 1,14,12:SETCOLOR 2
,11,0:SETCOLOR 3,3,0:SETCOLOR 4,8,4
120 POKE 752,1
130 ? " ! The Bumble Bee ∂":? " v":? "
     from The Legend of Tsar Saltan":?
 "                    N. Rimsky-Korsakoff
";
131 REM Line 130: Replace ! with
<CTRL>B. Replace ∂ with <CTRL>A
140 COLOR 215:FOR Z=14 TO 19:PLOT 0,Z:
DRAWTO 19,Z:NEXT Z
150 FOR Z=0 TO 9:IF Z=5 OR Z=7 THEN NE
XT Z
160 COLOR 130:PLOT Z,12:COLOR 214:PLOT
 Z,13:NEXT Z
170 COLOR 4:PLOT 14,8:COLOR 3:PLOT 15,
8:COLOR 5:PLOT 16,8:COLOR 3:PLOT 14,9:
DRAWTO 16,9
```

```
180 COLOR 87:FOR Z=11 TO 15 STEP 2:PLO
T 12,Z:DRAWTO 18,Z:NEXT Z:COLOR 4:PLOT
 13,9:COLOR 5:PLOT 17,9
190 COLOR 3:FOR Z=10 TO 14 STEP 2:PLOT
 12,Z:DRAWTO 18,Z:NEXT Z:COLOR 4:PLOT
12,10:COLOR 5:PLOT 18,10
```

Sound and graphics displays are integrated in line 200. Line 220 erases the bee before returning to line 200 for the next plot position.

```
200 COLOR 1:READ UN:PLOT UN/9,12-UN/15
:SOUND 0,UN,10,8
210 FOR Z=1 TO 15:NEXT Z
220 SOUND 0,0,0,0:COLOR 0:PLOT UN/9,12
-UN/15:GOTO 200
```

Data for redefined characters.

```
1000 DATA 0,0,0,0,0,0,0,0
1001 DATA 216,81,115,214,116,30,31,14
1002 DATA 40,146,146,214,254,124,56,16
1003 DATA 170,170,170,170,170,170,170,
170
1004 DATA 0,2,10,42,42,170,170,170
1005 DATA 0,128,160,168,168,170,170,17
0
1010 DATA 17,19,151,222,80,112,16,16
1011 DATA 255,255,255,255,255,255,255,
255
```

Data for music.

```
1040 DATA 96,102,108,114,108,114,121,1
28,96,102,108,114,108,114,121,128
1060 DATA 96,102,108,114,121,128,136,1
44,153,144,136,128,121,114,108,102
1080 DATA 96,102,108,114,121,91,96,102
,96,102,108,114,121,114,108,102
1100 DATA 96,102,108,114,121,91,96,102
,96,102,108,114,121,114,108,102
1120 DATA 96,102,108,114,108,114,121,1
28,121,114,108,102,96,91,96,102
1140 DATA 96,102,108,114,108,114,121,1
28,121,114,108,102,96,85,81,76
1160 DATA 72,76,81,85,91,68,72,76,72,7
6,81,85,91,85,81,76
1180 DATA 72,76,81,85,91,68,72,76,72,7
6,81,85,91,85,81,76
1200 DATA 72,76,81,85,81,85,91,96,91,8
5,81,76,72,68,72,76
1220 DATA 72,76,81,85,91,96,102,96,91,
85,81,76,72,68,72,76
1240 DATA 72,144,144,144,144,144,144,1
44,136,153,136,153,136,153,136,153
1260 DATA 144,144,144,144,144,144,144,
144,136,153,136,153,136,153,136,153
1280 DATA 144,136,144,153,144,136,144,
153,144,136,144,153,144,136,144,153
1300 DATA 144,136,128,121,114,121,128,
136,144,136,128,121,114,108,102,96
1320 DATA 108,108,108,108,108,108,108,
108,102,114,102,114,102,114,102,114
1340 DATA 108,108,108,108,108,108,108,
108,102,114,102,114,102,114,102,114
1360 DATA 108,102,108,114,108,102,108,
114,108,102,108,114,108,102,108,114
1380 DATA 108,102,96,91,85,91,96,102,1
08,102,96,91,85,81,76,72
1400 DATA 53,57,60,64,68,50,53,57,53,5
7,60,64,68,64,60,57
1420 DATA 53,57,60,64,60,64,68,72,68,6
4,60,57,60,57,53,50
1440 DATA 47,50,53,57,53,57,60,64,60,6
4,68,72,76,81,85,91
1460 DATA 96,91,96,102,96,91,96,102,96
,91,96,102,96,91,96,102
1480 DATA 96,91,96,102,96,91,96,102,96
,91,96,102,96,91,96,102
1500 DATA 47,47,47,47,47,47,60,60,72,7
2,91,91,72,72,60,60
1520 DATA 47,47,47,47,47,47,60,60,72,7
2,91,91,72,72,60,60
1540 DATA 47,47,47,47,47,47,47,47,47,4
7,47,47,47,47,47,47
1560 DATA 96,96,96,96,96,91,85,81,76,7
2,68,64,60,57,53,50
1580 DATA 47,50,53,57,60,45,47,50,47,5
0,53,57,60,57,53,50
1600 DATA 47,50,53,57,60,45,47,50,47,5
0,53,57,60,57,53,50
```

```
1620 DATA 47,50,53,57,53,57,60,64,60,5
7,53,50,47,45,47,50
1640 DATA 47,50,53,57,53,57,60,64,60,5
7,53,50,47,42,40,37
1660 DATA 35,37,40,42,45,33,35,37,35,3
7,40,42,45,42,40,37
1680 DATA 35,37,40,42,45,33,35,37,35,3
7,40,42,45,42,40,37
1700 DATA 35,37,40,42,40,42,45,47,45,4
2,40,37,35,33,35,37
1720 DATA 35,37,40,42,47,45,42,40,37,3
5,33,31,29,27,25,24
1740 DATA 23,24,25,27,29,21,23,24,23,2
4,25,27,29,27,25,24
1760 DATA 23,24,25,27,29,21,23,24,23,2
4,25,27,29,21,23,24
1780 DATA 85,81,76,72,68,64,60,57,53,5
7,60,64,60,64,68,72
1800 DATA 76,72,68,64,60,57,53,50,47,4
5,47,50,47,45,47,50
1820 DATA 47,81,76,72,68,64,60,57,53,5
7,60,64,60,64,68,72
1840 DATA 76,72,68,64,60,57,53,50,47,4
5,47,50,47,42,40,37
1860 DATA 35,37,40,42,40,42,45,47,45,4
7,50,53,57,60,64,68
1880 DATA 72,76,81,85,81,85,91,96,91,9
6,102,108,114,121,128,136
1900 DATA 144,136,144,153,144,136,144,
153,144,136,144,153,144,128,121,108
1920 DATA 96,91,96,102,96,91,96,102,96
,91,96,102,96,85,81,76
1940 DATA 72,81,85,91,96,102,108,114,1
21,128,136,153,144,136,128,121
1960 DATA 114,108,102,96,91,85,81,76,7
2,68,64,60,57,53,50,47
1980 DATA 35,35,35,35,35,35,35,35,72,7
2,72,72,96,96,96,96
```

Plot the bee's final position, then restart the program.

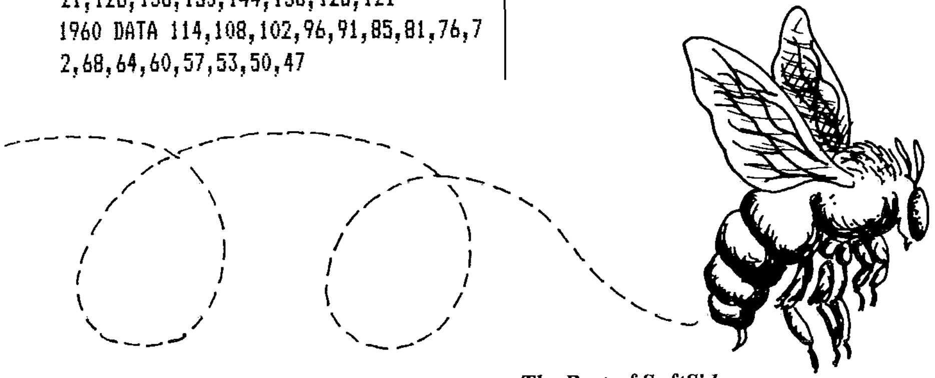
```
19999 SOUND 0,144,10,8:COLOR 1:PLOT 15
,7:RESTORE :FOR Z=1 TO 64:READ X:NEXT
Z:TRAP 19999:SOUND 0,0,0,0
20000 FOR Z=1 TO 400:NEXT Z:GOTO 110
```

Adjust the screen display to permit the use of three graphics modes in the title display.

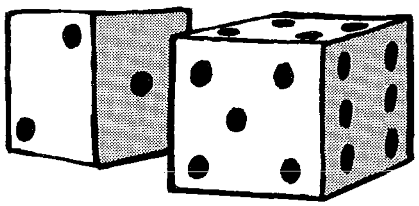
```
30200 GRAPHICS 0:SETCOLOR 2,8,4:SETCOL
OR 4,8,4:UB=PEEK(560)+PEEK(561)*256+4:
POKE UB-1,70:POKE UB+2,7:POKE UB+3,7
30210 FOR UZ=4 TO 8:POKE UB+UZ,6:NEXT
UZ:POKE UB+22,65:POKE UB+23,PEEK(560):
POKE UB+24,PEEK(561):SETCOLOR 3,3,0
30220 SETCOLOR 1,5,0:RETURN
```

Allocate memory space for the redefined character set, then read the data for the redefinitions.

```
30300 POKE 106,PEEK(106)-5:RETURN
30310 UC=256*(PEEK(106)+1):FOR UZ=0 TO
 1023:POKE UC+UZ,PEEK(57344+UZ):NEXT U
Z
30320 POKE 756,UC/256:FOR UZ=512 TO 55
9:READ UY:POKE UC+UZ,UY:NEXT UZ
30330 FOR UZ=944 TO 959:READ UY:POKE U
C+UZ,UY:NEXT UZ:RETURN
```



Melody Dice



by Gary Cage
Atari version by Rich Bouchard

Melody Dice **is a musical game program for an Atari with 24K RAM.**

This is a computerized adaptation of a game in which there are 60 flash cards, each containing one measure of a song by Scott Joplin. Ten of these cards are picked at random by rolling a pair of dice five times. The numbers rolled are added to the roll number to determine the numbers of the cards. You then take these ten cards to the piano, and play the "composition," which Scott Joplin never wrote, but is nonetheless his music.

I adapted this format to the computer. The program consists of a dice routine to choose the "cards" at random, a music routine to play them, a graphics routine to display the music as it is being played, and routines to save or recall a created song to or from disk or tape. In this case, the cards consist of strings of numbers in groups of six. The first three numbers represent the note to be played, and the second three represent the note's duration.

The program includes a Machine Language routine, which draws the high-resolution notes and ensures proper note duration. Atari BASIC was too slow to perform this function, so *SoftSide's* Alan J. Zett worked out this special routine.

Variables

A(i): Holds cards corresponding to die A's throw. This is the only variable used in the playback routine.
AA(i): Holds A cards while B is being transferred to A.
AB: Lines up the five cards for each die.
AN, AN$: Player's response.
B(i): Holds cards corresponding to die B's throw.
BH, BV: Horizontal and vertical locations to print numbers on die B.
C: Card number (1-60).
DA: Number thrown on die A.
DB: Number throw on die B.
DK: Code that determines to which line the error-trap routine ought to return. DK = 0 means go back to the SAVE routine; DK = 1 means go back to the RECALL routine.
ER: Error code.
H: Variable for HTABs.
HC: Horizontal clear variable for dice routine.
I, II: General counter variables.
KEY: Value of pressed key.
L: Note length.
LL: Cumulative length of notes. Determines when a measure is full.
MO: Increments the column in which numbers thrown on the dice are displayed.
MS: Measure number.
N: Note pitch.
NA$: Program author's name; used for title page.
NN, NN%: Hold a value to add notes to the Y axis when drawing notes; this determines where the note is drawn.
S$: Name given to a saved song.
SW: Switch to keep track of whether a FOR loop has been used more than once. (0 = no, 1 = yes).

T: Toss number.
V: Variable for HTABs.
VA: Value to be POKEd into memory.
VC: Vertical clear variable for dice routine.
X, Y: x- and y-axis values.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS        'Melody  Dice'       SS
SS      Author: Gary Cage      SS
SS Translator:  Rich Bouchard SS
SS      Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Main program. This is essentially a menu.

```
5 CLR :OPEN #2,4,0,"K:"
10 GOTO 20
12 SOUND 0,N,10,10
14 FOR ZZ=1 TO L/2:NEXT ZZ:SOUND 0,0,0
,0:FOR ZZ=1 TO 5:NEXT ZZ:RETURN
20 GRAPHICS 0
30 GOSUB 1000
40 GOSUB 2000
50 ? :? "Would you like instructions (
Y/N) ? ";
60 GET #2,AN:AN$=CHR$(AN)
70 IF AN$="N" THEN 100
80 IF AN$<>"Y" THEN 60
90 GOSUB 2500
100 GRAPHICS 0
110 POSITION 6,3:? "*** Would you like
 to:"
120 ? :? ,"1) Play a song in memory?"
130 ? ,"2) Create a new song?"
140 ? ,"3) Play a song from media"
142 ? ,"4) Quit?"
150 POSITION 14,13:? "Number-> ";
160 GET #2,AN
170 AN=AN-48:IF AN<1 OR AN>4 THEN 160
175 IF AN=1 THEN GOSUB 4000:GOTO 100
180 IF AN=2 THEN AB=1:GOTO 3000
190 IF AN=3 THEN GOTO 6000
200 GRAPHICS 0:PRINT "Okay ... Bye":EN
D
```

Initialize C$(I) and load note routine into memory.

```
1000 REM
1090 DIM C$(2000),C(61),Z$(120),NN(21)
,AN$(13),X$(60)
1092 DIM A(10),AA(10),B(10)
1094 READ Z$:FOR T=1 TO LEN(Z$) STEP 2
:A=ASC(Z$(T,T))-48:IF A>11 THEN A=A-7
1096 A=A*16+ASC(Z$(T+1,T+1))-48:IF Z$(
T+1,T+1)>"@" THEN A=A-7
1098 X$(LEN(X$)+1)=CHR$(A):NEXT T
1100 RESTORE 10000:FOR I=0 TO 60
1110 READ Z$:C(I)=LEN(C$)+1:C$(LEN(C$)
+1)=Z$
1120 NEXT I:C(61)=LEN(C$)+1
1140 NN(0)=-4:NN(1)=-2:NN(2)=0:NN(3)=0
:NN(4)=2:NN(5)=4:NN(6)=6:NN(7)=6
1142 NN(8)=8:NN(9)=0:NN(10)=10:NN(11)=
10:NN(12)=12:NN(13)=12:NN(14)=0
1150 NN(15)=14:NN(16)=14:NN(17)=0:NN(1
8)=16:NN(19)=0:NN(20)=18:NN(21)=18
1160 RETURN
```

Title page. For the music, the routine divides C$(0) into groups of six characters. The first three characters of each group provide the value for N (note); the last three, for L (length).

```
2000 GRAPHICS 2:POKE 752,1:SETCOLOR 2,
0,0
2020 POSITION 6,3:PRINT #6;"MELODY":PR
INT #6:PRINT #6;"      dice"
2050 REM
2060 PRINT "    By G. Cage & R. Boucha
rd";
2070 A=PEEK(560)+PEEK(561)*256:MEM=PEE
K(A+4)+PEEK(A+5)*256+200
2080 FOR II=1 TO C(1)-1 STEP 6:N=VAL(C
$(II,II+2)):L=VAL(C$(II+3,II+5)):SOUND
 0,N,10,10
2090 A=USR(ADR(X$),MEM,L/8):NEXT II:SO
UND 0,0,0,0
2100 FOR I=1 TO 500:NEXT I
2110 GRAPHICS 0:RETURN
```

Instructions.

```
2500 GOSUB 2900
2510 FOR I=1 TO 500:NEXT I
2530 ? "  Imagine if you will, that th
ere are 60 cards stored somewhere in A
tari,"
2532 ? "each containing portions of mu
sic     stolen from Scott Joplin."
2540 ? :? "  Now what if you were to j
umble up   those cards in a random ord
er and play";
2542 ? "them on an instrument.  Would
they    sound as good as the originals
?"
2550 ? :? "Probably not.  But then aga
in, who    knows?  At any rate, you wo
uld have"
2552 ? "yourself an original compositi
on      inspired by one of the greats
in      music."
2560 ? :? "And if you didn't like it,
you could  mix up the cards once more
and have"
2562 ? "something completely different
."
2570 REM
2580 POSITION 6,23:? "< Press RETURN t
o continue >";
2590 GOSUB 2800
2600 GOSUB 2900
2610 PRINT "  Can't play an instrument
 or read   music?  Well, that's where
 Atari comes";
2612 ? "in.  You will be shown a pair
of dice.Press the space bar to stop th
em from"
2620 ? "spinning, and the numbers on y
our     throw will be displayed.  This
 is done for a total of 5 tries."
2630 ? :? "  Each number of the die co
rresponds  to a particular card in mem
ory depen-"
2632 ? "ding on the time it was thrown
 (1st   try, 2nd try, etc.).  Atari"
2640 ? "assembles these together and y
ou are  asked if you want to hear the
final   product, or try again."
2650 ? :? "  If you wish to have it pl
ayed, the  music will be accompanied b
y its"
2652 ? "musical notation (of sorts)."
2660 GOSUB 2800
2670 GOSUB 2900
2680 ? "After listening to your compos
ition,  you will be asked if you wish
to save"
2682 ? "it on disk, try again, or quit
        altogether."
2690 ? :? "That's all there is to it."
2700 GOSUB 2800
2710 RETURN
2800 GET #2,AN
2810 IF AN<>155 THEN 2800
2820 RETURN
2900 GRAPHICS 0:? "          * INSTRUCT
IONS *":? :? :RETURN
```

Dice routine. This displays the dice, and allows you to stop their spinning.

```
3000 GRAPHICS 0:POKE 752,1
3010 FOR H=7 TO 15:POSITION H,0:? CHR$
(160);:POSITION H+11,7:? CHR$(160);:NE
XT H
3020 FOR V=0 TO 8:POSITION 15,V:? CHR$
(160);:POSITION 26,7+V:? CHR$(160);:NE
XT V
3030 FOR H=15 TO 7 STEP -1:POSITION H,
8:? CHR$(160);:POSITION 11+H,15:? CHR$
(160);:NEXT H
3040 FOR V=8 TO 0 STEP -1:POSITION 7,V
:? CHR$(160);:POSITION 18,7+V:? CHR$(1
60);:NEXT V
3050 V=10:H=9:POSITION H,V:? CHR$(160)
;:POSITION H+4,V:? CHR$(160);:POSITION
 H+11,V+7:? CHR$(160);
3060 POSITION H+15,V+7:? CHR$(160);
3070 GOSUB 3360
3080 GOSUB 3430
3090 T=0:DA=INT(RND(0)*6)+1:DB=INT(RND
(0)*6)+1:MO=0
3100 GOSUB 3470
3110 GOSUB 3500
3120 H=8:V=3
3130 BH=0:BV=0:ON DA GOSUB 3190,3210,3
230,3250,3270,3290
3140 BH=11:BV=7:ON DB GOSUB 3190,3210,
3230,3250,3270,3290
3150 REM
3160 GOSUB 3600
3170 IF T>=5 THEN FOR I=1 TO 500:NEXT
I:GOTO 3800
3180 GOTO 3100
3190 POSITION H+3+BH,V+1+BV:? "∂";
3200 RETURN
3210 POSITION H+BH+1,V+BV-1:? "∂";:POS
ITION H+5+BH,V+3+BV:? "∂";
3220 RETURN
3230 FOR I=0 TO 4 STEP 2:POSITION H+I+
BH+1,V+I+BV-1:? "∂";:NEXT I
3240 RETURN
3250 GOSUB 3310
3260 RETURN
3270 GOSUB 3310:POSITION H+3+BH,V+1+BV
:? "∂";
3280 RETURN
3290 GOSUB 3310:POSITION H+BH+1,V+BV+1
:? "∂";:POSITION H+5+BH,V+BV+1:? "∂";
3300 RETURN
3310 POSITION H+BH+1,V+BV-1:? "∂";:POS
ITION H+5+BH,V+BV-1:? "∂";:POSITION H+
5+BH,V+3+BV:? "∂";
3320 POSITION H+BH+1,V+3+BV:? "∂";:RET
URN
3330 FOR VC=2 TO 6 STEP 2:POSITION 9,V
C:? "     ";:POSITION 9+BH,VC+BV:? "
   ";
3340 NEXT VC
3350 RETURN
3360 REM
3370 POSITION 23,0:? "* DICE *"
3380 POSITION 17,1:? "You are given fi
ve";:POSITION 17,2:? "tries at the dic
e.";
3390 POSITION 17,4:? "Press Space Bar
to";:POSITION 17,5:? "stop them spinni
ng";
3400 POSITION 28,15:? "> Try #   <"
3410 REM
3420 RETURN
3430 REM
3440 POSITION 2,20:? "DIE A:"
3450 POSITION 2,22:? "DIE B:"
3460 RETURN
3470 T=T+1
3480 POSITION 36,15:? T;
3490 RETURN
3500 V=10:H=11:POSITION H,V:PRINT DA:P
OSITION H+11,V+7:PRINT DB
3510 GOSUB 3530
3520 RETURN
3530 KEY=PEEK(764):IF KEY=33 THEN 3560
3540 DA=INT(RND(1)*6)+1:DB=INT(RND(1)*
6)+1
3550 POP :GOTO 3500
3560 GOSUB 3330
3570 POKE 764,255:POSITION MO+9,20:? D
A;:POSITION MO+9,22:? DB;
```

```
3580 MO=MO+3
3590 RETURN
3600 C=DA*T+(6-DA)*(T-1)
3610 A(AB)=C
3620 C=(DB*T+(6-DB)*(T-1))+30
3630 B(AB)=C
3640 AA(AB)=A(AB)
3650 AB=AB+1
3660 RETURN
3800 GRAPHICS 0:POSITION 6,5:? "--> Do
 you wish to play this":POSITION 10,6:
? "song or try again?"
3810 POSITION 18,8:? "Type Play or":PO
SITION 23,9:? "Again."
3820 POSITION 25,10:? "--> ";
3830 GET #2,AN:AN$=CHR$(AN)
3840 IF AN$="A" THEN AB=1:GOTO 3000
3850 IF AN$<>"P" THEN 3830
3860 GOSUB 4000
3870 FOR I=1 TO 1000:NEXT I
3880 GRAPHICS 0:? "Play the same song
again ? ";
3890 GET #2,AN
3900 AN$=CHR$(AN):IF AN$="N" THEN 5000
3910 IF AN$<>"Y" THEN 3890
3920 GOTO 3860
```

Music routine. This begins with a GOSUB to the routine that draws the musical staff. The routine at line 4900 calculates where to draw the notes. The note length is continually added to LL, and when this variable's value reaches or exceeds 255 (four counts), a measure bar is drawn.

```
4000 GRAPHICS 0:GOSUB 4500:SW=0
4010 FOR I=1 TO 4
4020 GOSUB 4300
4030 NEXT I
4040 FOR I=1 TO 3
4050 GOSUB 4300
4060 NEXT I
4070 I=5
4080 GOSUB 4300
4090 IF SW=1 THEN 4120
4100 SW=1:FOR I=1 TO 5:A(I)=B(I):NEXT
I
4110 GOTO 4010
4120 FOR I=1 TO 5
4130 A(I)=AA(I)
4140 NEXT I
4150 FOR I=1 TO 3
4160 GOSUB 4300
4170 NEXT I:I=5
4180 GOSUB 4300
4190 RETURN
4300 FOR II=C(A(I)) TO C(A(I)+1)-1 STE
P 6
4310 N=VAL(C$(II,II+2))
4320 L=VAL(C$(II+3,II+5))
4330 IF N>1 THEN SOUND 0,N,10,10
4340 GOSUB 4900
4350 NEXT II
4360 RETURN
4500 GRAPHICS 24:COLOR 1:A=PEEK(560)+P
EEK(561)*256:MEM=PEEK(A+4)+PEEK(A+5)*2
56
4520 Y=10:X=10
4530 FOR Y=10 TO 170 STEP 39:FOR I=0 T
O 20 STEP 5
4540 PLOT X,Y+I:DRAWTO X+289,Y+I
4550 NEXT I:NEXT Y
4630 X=16:Y=6
4640 FOR I=0 TO 170 STEP 39
4650 PLOT X,Y+I:DRAWTO X,Y+I+28
4660 REM *****
4670 NEXT I
4690 LL=0:MS=1:X=27
4700 RETURN
4900 IF N=1 THEN A=USR(ADR(X$),MEM,L/1
0-2):GOTO 4930
4902 NN=INT(NN(N/5.7-10)*1.25+0.5):A=U
SR(ADR(X$),MEM+(Y+NN+2)*40+INT(X/8),L/
10-2)
4920 IF N=57 THEN PLOT X-4,Y+NN+4:DRAW
TO X+5,Y+NN+4
4930 SOUND 0,0,0,0:X=X+8:LL=L+LL
4950 IF LL>=255 THEN LL=0:PLOT X,Y+4:D
RAWTO X,Y+24:X=X+8:MS=MS+1
```

```
4960 IF MS>4 THEN MS=1:X=27:Y=Y+39
4970 RETURN
```

Save music to disk or tape.

```
5000 GRAPHICS 0
5010 POSITION 8,2:? "Would you like to
 save this song      (Y/N) ?";
5020 GET #2,AN:AN$=CHR$(AN)
5030 IF AN$="N" THEN 100
5040 IF AN$<>"Y" THEN 5020
5050 POSITION 8,7:? "On tape or disk (
T/D) ?";
5060 GET #2,AN:AN$=CHR$(AN)
5070 IF AN$="T" THEN 5260
5080 IF AN$<>"D" THEN 5060
5090 REM
5100 DK=0
5110 POSITION 14,14:? CHR$(253);"Name
->";
5120 INPUT AN$:Z$(1,2)="D:":Z$(3)=AN$
5130 REM
5140 REM
5150 OPEN #1,8,0,Z$
5190 FOR I=1 TO 5
5200 PRINT #1;A(I)
5210 PRINT #1;B(I)
5220 PRINT #1;AA(I)
5230 NEXT I
5240 CLOSE #1
5250 GOTO 100
5260 GRAPHICS 0:? "      * Store to ta
pe *"
5270 POSITION 2,4:? "1. Make sure your
 tape is ready to   record."
5280 ? :? "2. Start recording and then
 push      'Return'.  Atari will retur
n you to   the program when finished"
5290 ? :? "3. If you get an error mess
age, set uptape again and type 'GOTO 5
260'"
5300 OPEN #1,8,0,"C:"
5310 GOTO 5190
```

Recall from tape or disk.

```
6000 GRAPHICS 0
6010 POSITION 14,7:? "From tape or dis
k (T/D) ?";
6020 GET #2,AN:AN$=CHR$(AN)
6030 IF AN$="T" THEN 6210
6040 IF AN$<>"D" THEN 6020
6050 REM
6060 DK=1
6070 POSITION 14,14:? CHR$(253);"Name
->";
6080 INPUT AN$:Z$(1,2)="D:":Z$(3)=AN$
6090 REM
6100 REM
6110 OPEN #1,4,0,Z$
6130 FOR I=1 TO 5
6140 INPUT #1,Z:A(I)=Z
6150 INPUT #1,Z:B(I)=Z
6160 INPUT #1,Z:AA(I)=Z
6170 NEXT I
6180 CLOSE #1
6190 GRAPHICS 0:GOSUB 4000
6200 GRAPHICS 0:GOTO 100
6210 GRAPHICS 0:? "      * Recall from
 tape *"
6220 POSITION 2,4:? "1. Make sure your
 tape is set up and  ready at your mus
ic program."
6230 ? :? "2. Start recorder on play a
nd then    push 'Return'."
6240 ? :? "3. If all goes well, Atari
will returnto the program."
6250 ? :? "4. If you get an error mess
age, re-runprogram and try again."
6260 OPEN #1,4,0,"C:"
6270 GOTO 6130
```

Data for note-drawing routine.

```
8000 DATA 686885CC6885CB1BA000A97E91CB
986928A8A97E91CB986928A8A97E91CB986928
A8A97E91CB6868A8A514C514F0FCB8D0F760
```

Data for music "cards."

```
10000 DATA 1440640960641440641140640961281530640960641530641280640962551
10010 DATA 068032064064068032064064068032064032,0640320640640680320640320640
32057032064032
10020 DATA 076032081032076032064064096032076032064032,0640320640640730320760
32081032076032064032
10030 DATA 064064068032064064068032064064,0640320570320640320760320720320860
64076032
10040 DATA 064032076032064032057064064064096032,076064086032096096086064
10050 DATA 096032086032076032128064076032086032096032,0761601230320960320760
32
10060 DATA 076064086032096064064032076032096032,0760640640320760640810320760
64
10070 DATA 086064064032086064064032086064,0010320860320570320720320640320570
32064032072032
10080 DATA 072032072064076032072032072032076032086032,0720320570640640320720
32076032072032064032
10090 DATA 072064072032072064072032086064,064096068032064032057032064064
10100 DATA 064160076032072032068032,06
4064102032096032086032076032072032064032
32
10110 DATA 064255,064255
10120 DATA 064096068032064128,06406406
4032064032064064064064
10130 DATA 096128096064001064,07606408
6064096128
10140 DATA 096064001064096064001064,09
6064001064096064001064
10150 DATA 096064001064128032114032102
032096040001064,096064128064096128
10160 DATA 102064102032128064102032128
064,102032171032138032114064128032114064
```

```
10170 DATA 102032108032102032128064128
032108032102032,0010321280320760320860
65076032086064
10180 DATA 001032128032076032102032086
032076032086032102032,1020321020641140
32128032136032128032102032
10190 DATA 128032102032086032064032064
064086064,103032171032128032102160
10200 DATA 128032102064096032086064064
032086032,0760320860640760320860320760
32086064
10210 DATA 128032102032076032086096086
064,0860320860640960321020321080321020
32086032
10220 DATA 096032068064096064068032096
064,096064114064076032086064096032
10230 DATA 096064086032096064102032096
064,0960641140320960641140320960321140
32
10240 DATA 001032086032068032086032076
032068032076032086032,0960321140321020
32096064076032086064
10250 DATA 114032114064121032114064108
064,096032096064076032086032096032086064
10260 DATA 096032096064102032096032096
032102032114032,1210321140641020320760
64096064
10270 DATA 096128086032086064096032,08
6128086064096064
10280 DATA 102064001064128064001064,10
2160086032076032086032
10290 DATA 102064064032064032064064064
064,102064001064128064001064
10300 DATA 102192064064,10206410206408
6064064064
```

Music Programmer

by John Rush Elkins

Music Programmer **is a music editing program for a 24K Atari (32K with disk).**

The use of this program will be explained while progressing through an example. The musical scale "DO, RE, MI" is shown being entered as a demonstration program, with eighth notes, in four voices. The last note in the measure is shown being entered into the Enter Music routine.

MUSIC PROGRAMMER

NOTES AND RESTS ARE ENTERED AND EDITED MEASURE BY MEASURE IN 1 TO 4 VOICES USING A ROUTINE CALLING FOR NOTE, DURATION, OCTAVE, AND LOUDNESS. MEASURE LENGTH IS DETERMINED FROM THE TIME SIGNATURE. TIED NOTES, DOTTED NOTES, AND TRIPLETS ARE ENTERED WITH SUBSCRIPTS. TIED MEASURES ARE COMBINED INTO ONE LARGE MEASURE. SHARPS AND FLATS ARE ENTERED WITH THEIR NOTES SINCE THERE ARE NO KEY SIGNATURES. EACH MEASURE MAY BE PLAYED FOR EDITING BEFORE SAVING.

EACH MEASURE IS THEN SAVED BY TRANSFERRING TO DATA STATEMENTS WHICH MUST BE ENTERED INTO THE PROGRAM.

PRESS RETURN TO CONTINUE.

Music which can be enjoyed on its own or entered into other BASIC programs. No more than minimal musical and programming skills are required. However, the ability to manipulate the editing features of the Atari is required to enter the computer-generated DATA statements.

The *Music Programmer* utilizes musical measures as its "bookkeeper" to insure that the notes start and stop at the proper place. If a note needs to be continued into the next measure, the two measures are tied (⁀) by the note into a double measure which is entered as one large measure.

WHAT IS THE MAXIMUM NUMBER OF NOTES — UP TO 16 — YOU WILL NEED TO ENTER INTO ANY SINGLE VOICE IN ANY MEASURE? MEASURES TIED TOGETHER WITH CONTINUING NOTES ARE TREATED AS ONE LARGE MEASURE.
?12

This value (V0 in line 818) is required to DIMension the arrays to hold the notes. The maximum number of notes to be entered can be increased (lines 818, 819, 814); but the smaller the value, the more memory is made available for saving the notes in DATA statements.

HOW MANY VOICES (1 TO 4) WILL YOU NEED
?4

The number of voices needed is determined by the number of notes that must be played at any one time. Use of only one voice, playing the melody (top) line of notes, will require less memory and can be entered faster, but will not give the fullness of sound of which the Atari is capable. Musical notation often does not show rests for all voices, so be sure to take care to fill out blank spaces in the music with rests. It may also be necessary to add rests at the beginning of the first measure and at the end of the last measure. The computer requires that a measure of music be entered in each voice selected.

TIME SIGNATURE?
WHICH? 1 FOR 4/4
2 FOR 3/4 OR 6/8
3 FOR 2/4
4 FOR 2/2
?1

The time signature gives the measure length with the 4/4 (or C) and 2/2 (or ₵) times being equivalent in length to a whole note (four counts). The 2/2 time, as used in the *Music Programmer*, plays twice as fast as the 4/4 time. The 3/4 and 6/8 times are equivalent in length to three-fourths of a whole note (three counts) and the 2/4 time to a half note (two counts).

TEMPO? FAST TO SLOW
WHICH? 8 9 10 11 12

WITH 9 OR 11: NO DOTTED 16th NOTES AND NO 16th OR DOTTED 8th NOTES WITH 2/2 TIME.

WITH 10 OR 12: NO 16th NOTES WITH 2/2 TIME.

WITH 9 OR 12: TRIPLETS ARE ENTERED NORMALLY WITH TEMPOS 9 AND 12 AND AS TWO DOTTED AND ONE NORMAL NOTE OF THE NEXT FASTER SPEED WITH OTHER TEMPOS. NO TRIPLET 16th NOTES WITH 9.
?10

The tempo is precisely controlled by the computer with the values (8-12) representing the duration of a sixteenth note in 60ths of a second. Notes with durations of less than 8/60ths of a second are not possible because of the time required to change notes. The menu indicates which durations are excluded, either for being too fast or for giving fractional values with dotted and triplet notes. Care should be taken in selecting a tempo which will fit the music at hand. For music with triplets, you should try to use tempos 9 or 12. For example, triplet eighth notes (♪♪♪) with a total duration of a quarter note can be entered normally as three E3 notes in tempos 9 and 12, but must be entered as two dotted sixteenth notes (S.) and one sixteenth note (S) with other tempos.

WHAT IS THE TITLE OF YOUR SONG?
CENTER IN 11th SPACE
WITH FIRST LINE HERE
AND SECOND LINE NEXT.
USE UPPER AND lower
CASE & INVERSE video
? DO,RE
CONTINUE TITLE HERE.
? mi

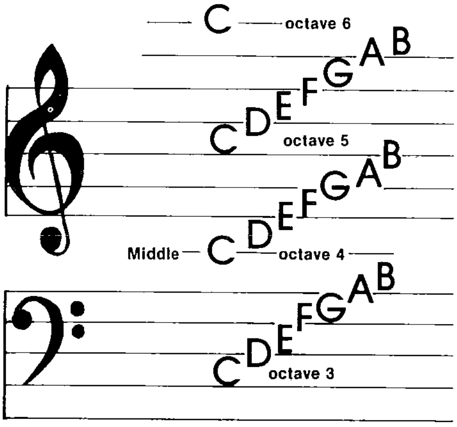


Figure 1
The range of music played by the Atari.

There are two lines provided for the title of your song. Your title may go on either or both of them. Each line should be centered on the eleventh space. Inverse video and lower-case letters are displayed as upper-case letters with different colors. Be sure to change back to normal upper-case letters.

ENTER TOTAL NUMBER OF COMBINED MEASURES.

HOW MANY? 1 FOR NO COMBINED MEASURES.
2, 3, 4 COMBINED MEASURES.

LIMIT 12 NOTES PER VOICE.
1

More measures can be combined by increasing V0 in lines 819 and 814 so that more notes can be entered, and increasing CM in lines 997 and 998; but such an increase would require more memory to reserve array space, and it would be more difficult to edit the measure in case of an entry error.

The following entries can be made for the *DO,RE,mi* demonstration program with eighth notes in four voices:

```
VOICE 0: C,E,SM,R,Q.,G,E,4,MF,R,Q.
VOICE 1: R,E,D,E,4,MF,R,Q.,A,E,4,MF,R,Q
VOICE 2: SM,E,E,4,MF,R,Q.,B,E,4,MF,R,E
VOICE 3: R,Q.,F,E,4,MF,R,Q.,C,E,5,MF
```

The last note entered in the measure is shown being entered into the Enter Music routine.

```
                ENTER MUSIC
THIS IS MEASURE 1
NUMBER OF AVERAGE MEASURES LEFT
= 59
VOICE 3: ENTRY 4
NOTE?
WHICH? AF,A,AS
                    BF,B
       C,CS
       DF,D,DS
       EF,E,ES
       F,FS
       GF,G,GS
       R FOR REST
ENTER SM FOR SAME NOTE
-R,Q.,OCTR,OFF
?C
```

The desired notes (A-G) and rests (R), with sharps (S) and flats (F), are entered here. The octave on the musical staff is entered from the menu after the next. The range of the Atari is over three full octaves, with one note in a fourth octave, as illustrated in Figure 1. Since the base range is not very low, it is often necessary to enter notes in a higher octave than that in which they are written.

Notice the prompts with the measure number, voice number and entry number which will help you keep track of where you are. The voice will change automatically when the measure is completed in that voice, as indicated by a bell. You should also pay attention to the bottom prompt, for there are many times when you can enter SM for the same note, especially the same octave and loudness.

```
VOICE 3: ENTRY 4, NOTE C

Total Duration  In  Measure  And  Program
VOICE 0 =           160           160
VOICE 1 =           160           160
VOICE 2 =           160           160
VOICE 3 =           140           140
DURATION OF NOTE OR REST?
WHICH? SIXTEENTH
       EIGHTH
       QUARTER
       HALF
       WHOLE
ADD . FOR DOTTED NOTES
ADD 3 FOR TRIPLET NOTES
ADD T FOR TIED NOTES
ENTER ED TO EDIT NOTE C
ENTER SM FOR SAME NOTE
-C,Q.,OCTR,OFF
?E
```

The notations for the allowed durations of notes are S (𝅘𝅥𝅯), E (♪), Q (♩), H (𝅗𝅥), W (𝅝), and of rests are S (𝄿), E (𝄾), Q (𝄽), H (𝄼), W (𝄻). Dotted notes such as "E." are 3/2 as long as a normal note and triplets such as E3 are 2/3 as long as a normal note. Triplets can only be entered with tempos 9 and 12; otherwise enter the triplet notes as two dotted and one normal note of the next faster duration. The duration of tied notes such as QT is followed up by the duration of the note to which it is tied, for example, "E.". No T should be added to the last of the tied notes. Tied notes are combined into one note with a total duration equal to the sum of the tied notes. Note that you cannot use SM to enter the note or duration after entering a tied note. The duration of the note entered with SM will be the duration of the tied note and not that of the note in the SM display, which shows only the last duration added to the tied note. You should also pay attention to the ED prompt which allows you to edit the note.

```
VOICE 3: ENTRY 4, NOTE C, DURATION E
OCTAVE? 4 GOES UP FROM MIDDLE C
WHICH? 3, 4, 5, 6
ENTER ED TO EDIT DURATION E
ENTER SM FOR SAME NOTE
-C,E,OCTR,OFF
?SM FOR SAME NOTE -C,E,OCTR,OFF
?5
```

The octave menu was discussed along with the note menu.

```
VOICE 3: ENTRY 4, NOTE C, DURATION
E, OCTAVE 5
LOUDNESS?
WHICH? PP,P,MP
       MF,F,FF
       OFF
NORMAL VALUES:
  MF FOR MELODY
  P FOR ACCOMPANIMENT
ENTER ED TO EDIT OCTAVE 5
?MF
```

There is much room for experimentation with loudness, but a value of MF for the melody (top) line of notes and P for the accompanying notes is a good starting place. The assigned values of loudness are PP=2, P=4, MP=6, MF=8, F=10, FF=12, and OFF=0. These can be adjusted from 1-15 in lines 1184 and 1188. Loudness can also be controlled with the Play Music routine (lines 30, 91) as explained below. Loudness can only be EDited by editing the entire measure.

If it should happen that you enter a wrong note and cannot get out of that voice, it is possible to BREAK and GOTO 5001 to reenter the entire measure in that voice.

```
        MEASURE FULL
PLAY MEASURE OR
EDIT MEASURE OR
SAVE MEASURE
WHICH?
ENTER 1 TO PLAY
      2 TO EDIT
      3 TO SAVE
```

The PLAY option lets you listen to the measure of music to check how it sounds before editing or saving it.

```
   EDIT MEASURE OF MUSIC
VOICE?
WHICH? 0,1,2,3
?0
EDIT VOICE 0
```

```
        SAVE MUSIC
MOVE CURSOR TO LINE NUMBER (LN)
AND PRESS RETURN TO ENTER INTO
PROGRAM. THEN TYPE CONT AND
PRESS RETURN TO CONTINUE.
LN
40 ?#6: ?#6;"DO, RE":?#6: ?#6;"mi"
STOPPED AT LINE 6514
CONT
```

The first time through the Save Music routine, the title can be entered into the program for the Play Music routine; and if there are fewer than four voices, certain "blank" lines are entered to remove lines not needed in the Play Music routine. Failure to enter the "blank" lines will result in incorrect play. If you make a mistake in manipulating the edit keys and fail to enter a line number correctly, you can return from the Play Music menu to reenter the line numbers.

```
LN
101 DATA 121,8,20,0,0,20,0,0,40,
0,0,60,0,0,60,108,8,20,0,0,60,96,8,20
102 DATA 0,0,60,91,8,20,81,8,20,
0,0,60,0,0,60,72,8,20,0,0,40,64,8,20
103 DATA 0,0,20,60,8,20
104 DATA 256,0,0
ENTER LINE NUMBERS AND CONTINUE
STOPPED AT LINE 7580
```

The notes have been placed in order of play in DATA statements which **must** be entered into the program for the Play Music routine. Each note READ by the Play Music routine requires three values: note, loudness and duration. Line 104 includes only the "256 flag" in Voice 0 which signals the END of play. Line 104 will be replaced by new DATA when saving the next measure and a new "256 flag" will END the play of both measures, etc. It may be helpful to keep a record of which line numbers correspond to which measures, so that you can manually make changes in the DATA statements if the music doesn't sound right.

```
            PLAY MUSIC
PLAY MUSIC OR
RETURN FOR NEW MEASURE OR
SAVE ON DISKETTE (CASSETTE) OR
RE-ENTER LINE NUMBERS
WHICH?
ENTER 1 TO PLAY
      2 TO RETURN
      3 TO SAVE
      4 TO RE-ENTER
?3
```

The Play Music menu allows you to play all the measures entered, return to enter a new measure, or re-enter line numbers. (The menu for the cassette version is included in parentheses and can be entered into the Music Programmer by replacing lines 31, 8450, 8452, 8453, 8454 and 8469 with the statements in lines 9031, 9450, 9452, 9453, 9454 and 9469.)

The music can be SAVEd to disk (or cassette) when completed or when all the memory is used up. The music, once saved, can be compiled with other music (if more than one SAVE is needed) by entering one and then the other, or can be entered into another BASIC program. Be careful not to have line numbers between 39 and 101 end up in the BASIC program since they will be wiped out upon entering the music program. Since the DATA statements are READ faster when they are closer to the beginning of the program, it may be advantageous to GOTO 500 at the program's start and then GOSUB 39 to PLAY MUSIC.

```
    SAVE ON DISKETTE (CASSETTE)
SAVE PLAY PROGRAM AND DATA WITH
THE INITIAL GROUP OF MEASURES BUT
ONLY DATA IN SUBSEQUENT SAVES.
INSERT BLANK DISKETTE! (Omitted in
cassette version)
ENTER LINE NUMBER AND CONTINUE
LN
8460 LIST "D:SONG1.
ENT",39,104(8460LIST"C",39,104)
STOPPED AT LINE 8453
PROGRAM BEING SAVED
(ON BELL, SET RECORDER TO RECORD
AND PRESS RETURN.)
ENTER LINE NUMBER TO CLEAR
MEMORY FOR NEXT GROUP OF
MEASURES.
LN
101
102
103
104
ENTER LINE NUMBERS AND CONTINUE
STOPPED AT LINE NUMBER 8465
```

For an additional SAVE, GOTO 8450 before entering the "blank" lines to free memory for a new group of measures.

```
LOAD OR ADD THIS MUSIC PROGRAM
AND ADDITIONAL DATA TO ANOTHER
PROGRAM WITH ENTER "D:SONG
#.ENT"
(ENTER "C")
```

On the second save of the Play program and DATA with the initial group of measures, change 101 to 39 when entering line 8460 with SONG2.ENT.

The Play Music routine (lines 30-91) and the Play/Edit Music routine (lines 700-765, with subroutines at lines 12-20) depend on the ability of the Atari to time itself to 1/60th of a second. The music is timed by POKEing 19 and 20 with 0 (lines 45 and 712), and PEEKing at the timer (PEEK 19*256 + PEEK 20 at lines 55 and 722) each time through the play cycle. All voices are sounded simultaneously (lines 50 and 720) each time through the play cycle, and are turned off with a perceptible pause (lines 81-84 and 751-754) each time there is a note change (lines 55-80 and 722-750). The notes are changed, voice by voice, whenever the time equals or exceeds the time when the previous note should stop playing.

Each voice may be played separately. For example, to listen to Voice 0 (the melody), LIST 50,80 and insert L1=0, L2=0 and L3=0 after the READ statements in lines 60, 70 and 80. Changes in loudness to values other than 0 can also be made.

Though very conservative parameters for the duration of the fastest note have been used in the Music Programmer, it is conceivable that with some songs there may be an occasional mis-ordering of notes because of the extra time required to go through each cycle of the note-ordering routine. If this happens, the "256 flag" probably will not be READ by Voice 0 and the program will "squawk" instead of ENDing. It can be corrected by hitting the BREAK key, entering END, GOing TO 30, and manually reordering the DATA, using the values for the notes given in the *Atari BASIC Reference Manual.*

VARIABLES

A$: Answer name.
B$: Bell character.
C: Column in array.
C$: Clear screen character.
CK0-CK3: Checks voices 0-3. 0=on, 1=off.
CM: Consecutive measures.
D: Duration number.
D0-D3: Duration of note in voices 0-3.
DL: Duration length.
DM: Duration of measure.
DT: Duration of tied note.
D$: Musical name of duration.
D(0,0)-D(3,0): Total duration of voices 0-3 in current measure.
D(0,1)-D(3,1): Total duration of voices 0-3.
ENT: Entry number.
H0-H3: Hold a note in voices 0-3. 0=hold, 1=no hold.
L: Loudness number.
L0-L3: Loudness of voices 0-3.
LN: Line number.
LN1: Line number at start of DATA.
LNF: Line number at end of DATA.
LNS: Line number at which "saving" should start.
L$: Musical name of loudness.
ML: Amount of memory left.
MN: Measure number.
MNT: Measure number total.
MV: Measure value.
MVA: Measure value average. Computed in line 1000.
MVT: Measure value total.
N: Note number.
NLV: Note limit value.
NM: Number of measures.
NV: Number of voices.
NS: Number of notes saved.
N0-N3: Notes being played by voices 0-3.
N$: Musical name of note.
N(N,O): Note value.
N0(V0,0)-N0(V3,0): Order in measure of notes V0-V3 in voices 0-3.
N0(V0,1)-N0(V3,1): Value of note in voices 0-3.
N0(V0,2)-N0(V3,2): Loudness of note in voices 0-3.
N0(V0,3)-N0(V3,3): Duration of note in voices 0-3.
O: Octave number.
ORD: Order of note in measure.
O$: Musical name for octave.
PES: Play, Edit, Save.
PRS: Play, Return, Save.
P0-P3: Turn off play in voices 0-3. 0=on, 1=off.
R: Row in array.
RELN: Re-enter line number.
R0-R3: Row in voices 0-3.
SN: Song number.
ST: Stop from re-entering title. 0=OK to enter, 1=not OK.
SV: Save in DATA statements. 0=off, 1=on.
T0-T3: Time to stop note in voices 0-3.

TM: Time of play.
TS: Time signature.
TITLE$: Title of song.
V: Voice number.
V0-V3: Number of note in voices 0-3.
V0E-V3E: Highest number of note in voices 0-3.
VN: Value of notes.
V$: Name of voice.
WT: Wait for message. Change this value in line 800 to change length of wait.

```
 SS SS SS SS SS SS SS SS SS SS SS
 SS                            SS
 SS          Atari  BASIC      SS
 SS     'Music Programmer'     SS
 SS  Author: John Rush Elkins  SS
 SS     Copyright (c) 1982     SS
 SS SoftSide Publications, Inc SS
 SS                            SS
 SS SS SS SS SS SS SS SS SS SS SS


10 GOTO 800
12 N0=N0(V0,1):IF N0=256 THEN N0(V0,0)
=ORD:POP :GOTO 760
14 L0=N0(V0,2):D0=N0(V0,3):T0=T0+D0:N0
(V0,0)=ORD:ORD=ORD+1:RETURN
16 N1=N1(V1,1):L1=N1(V1,2):D1=N1(V1,3)
:T1=T1+D1:N1(V1,0)=ORD:ORD=ORD+1:RETUR
N
18 N2=N2(V2,1):L2=N2(V2,2):D2=N2(V2,3)
:T2=T2+D2:N2(V2,0)=ORD:ORD=ORD+1:RETUR
N
20 N3=N3(V3,1):L3=N3(V3,2):D3=N3(V3,3)
:T3=T3+D3:N3(V3,0)=ORD:ORD=ORD+1:RETUR
N
30 ? "}":? "         **PLAY MUSIC**":?
:? "PLAY MUSIC OR":? :? "RETURN FOR NE
W MEASURE OR"
31 TRAP 30:? :? "SAVE ON DISKETTE OR":
? :? "RE-ENTER LINE NUMBERS":? :? "WHI
CH?":? :? "ENTER 1 TO PLAY"
32 ? "       2 TO RETURN":? "       3 TO
 SAVE":? "       4 TO RE-ENTER":INPUT P
RS
33 IF PRS=1 THEN 39
34 IF PRS=2 AND NM<=0 THEN ? CHR$(253)
:? " NO MORE MEASURES!":FOR W=1 TO WT:
NEXT W:GOTO 30
35 IF PRS=2 THEN 990
36 IF PRS=3 THEN 8450
37 IF PRS=4 THEN ST=0:LN1=RELN:GOTO 65
00
38 ? CHR$(253):GOTO 30
39 RESTORE 101:GRAPHICS 2+16:? #6:PRIN
T #6;"     PLay muSIc"
42 L0=0:N0=0:L1=0:N1=0:L2=0:N2=0:L3=0:
N3=0:P0=0:P1=0:P2=0:P3=0
45 TM=0:T0=0:T1=0:T2=0:T3=0:POKE 19,0:
POKE 20,0
50 SOUND 0,N0,10,L0:SOUND 1,N1,10,L1:S
OUND 2,N2,10,L2:SOUND 3,N3,10,L3
55 TM=PEEK(19)*256+PEEK(20):IF TM>=T0
THEN TM=T0:P0=1:READ N0,L0,D0:T0=T0+D0
:IF N0=256 THEN 90
60 IF TM>=T1 THEN TM=T1:P1=1:READ N1,L
1,D1:T1=T1+D1
70 IF TM>=T2 THEN TM=T2:P2=1:READ N2,L
2,D2:T2=T2+D2
80 IF TM>=T3 THEN TM=T3:P3=1:READ N3,L
3,D3:T3=T3+D3
81 IF P0=1 THEN SOUND 0,0,0,0:P0=0
82 IF P1=1 THEN SOUND 1,0,0,0:P1=0
83 IF P2=1 THEN SOUND 2,0,0,0:P2=0
84 IF P3=1 THEN SOUND 3,0,0,0:P3=0
85 GOTO 50
90 TRAP 91:FOR V=0 TO 3:SOUND V,0,0,0:
NEXT V:GOTO 30
91 END
700 ? CHR$(253):? "         **MEASURE
FULL**":? :? "PLAY MEASURE OR":? :? "E
DIT MEASURE OR":? :? "SAVE MEASURE"
701 TRAP 700:? :? "WHICH?":? :? "ENTER
 1 TO PLAY":? "       2 TO EDIT":? "
   3 TO SAVE":INPUT PES
702 IF PES=1 THEN 708
703 IF PES=2 THEN 5000
704 IF PES=3 THEN SV=1:GOTO 708
705 ? CHR$(253):GOTO 700
708 GRAPHICS 2+16:? #6:PRINT #6;"
PLay muSIc"
```

```
709 ? #6:? #6;TITLE1$:? #6:? #6;TITLE2
$
710 L0=0:N0=0:L1=0:N1=0:L2=0:N2=0:L3=0
:N3=0:P0=0:P1=0:P2=0:P3=0:ORD=0
712 V0=-1:V1=-1:V2=-1:V3=-1:T0=0:T1=0:
T2=0:T3=0:POKE 19,0:POKE 20,0
720 SOUND 0,N0,10,L0:SOUND 1,N1,10,L1:
SOUND 2,N2,10,L2:SOUND 3,N3,10,L3
722 TM=PEEK(19)*256+PEEK(20):IF TM>=T0
 THEN TM=T0:P0=1:V0=V0+1:GOSUB 12
725 IF H1=0 THEN 751
730 IF TM>=T1 THEN TM=T1:P1=1:V1=V1+1:
GOSUB 16
735 IF H2=0 THEN 751
740 IF TM>=T2 THEN TM=T2:P2=1:V2=V2+1:
GOSUB 18
745 IF H3=0 THEN 751
750 IF TM>=T3 THEN TM=T3:P3=1:V3=V3+1:
GOSUB 20
751 IF P0=1 THEN SOUND 0,0,0,0:P0=0
752 IF P1=1 THEN SOUND 1,0,0,0:P1=0
753 IF P2=1 THEN SOUND 2,0,0,0:P2=0
754 IF P3=1 THEN SOUND 3,0,0,0:P3=0
755 GOTO 720
760 FOR V=0 TO 3:SOUND V,0,0,0:NEXT V:
IF SV=1 THEN SV=0:GOTO 6500
765 GOTO 700
800 H1=0:H2=0:H3=0:V1=0:V2=0:V3=0:R0=3
:R1=0:R2=0:R3=0:DIM A$(3):WT=400
802 ? "}":? "          **MUSIC PROGRAMM
ER**":? :? "NOTES AND RESTS ARE ENTERE
D AND EDITED";
804 ? "MEASURE BY MEASURE IN 1 TO 4 VO
ICES   USING A ROUTINE CALLING FOR NOT
E,      DURATION,OCTAVE,";
806 ? " AND LOUDNESS. MEASURELENGTH IS
 DETERMINED FROM THE TIME    SIGNATURE
. TIED NOTES, DOTTED NOTES,"
807 ? "AND TRIPLETS ARE ENTERED WITH":
? "SUBSCRIPTS. TIED MEASURES ARE COMBI
NEDINTO ONE LARGE MEASURE.";
809 ? " SHARPS AND    FLATS ARE ENTERE
D WITH THEIR NOTES    SINCE THERE ARE
NO KEY SIGNATURES."
810 ? "EACH MEASURE MAY BE PLAYED FOR
EDITINGBEFORE SAVING."
812 ? :? "EACH MEASURE IS THEN SAVED B
Y":? "TRANSFERRING TO DATA STATEMENTS
WHICH MUST BE ENTERED ";
813 ? "INTO THE PROGRAM.":? :? "PRESS
RETURN TO CONTINUE.":INPUT A$
814 ? "}":? "WHAT IS THE MAXIMUM NUMBE
R OF NOTES-UPTO 16-YOU WILL NEED TO EN
TER INTO ANY"
816 ? "SINGLE VOICE IN ANY MEASURE? ME
ASURES TIED TOGETHER WITH CONTINUING N
OTES   ARE TREATED AS ONE ";
818 TRAP 814:? "LARGE MEASURE.":INPUT
V0:NLV=V0
819 IF V0<1 OR V0>16 THEN ? CHR$(253):
? " 1 TO 16 NOTES PER MEASURE!":FOR W=
1 TO WT:NEXT W:GOTO 814
820 TRAP 820:? "}":? "HOW MANY VOICES
(1 TO 4)WILL YOU NEED":INPUT NV
821 ON NV GOTO 826,825,824,823
822 ? CHR$(253):GOTO 820
823 H3=1:V3=V0:R3=R0
824 H2=1:V2=V0:R2=R0
825 H1=1:V1=V0:R1=R0
826 ? "}":? :? "TIME SIGNATURE?":? "WH
ICH? 1 FOR 4/4":? "        2 FOR 3/4 OR
 6/8":? "        3 FOR 2/4"
827 ? "        4 FOR 2/2":TRAP 826:INPU
T TS
828 IF TS<1 OR TS>4 THEN ? CHR$(253):G
OTO 826
829 IF TS=2 THEN TS=0.75
830 IF TS=3 THEN TS=0.5
833 ? "}":? "TEMPO? FAST TO SLOW":? "W
HICH? 8 9 10 11 12":? :? "WITH 9 OR 11
: NO DOTTED 16th NOTES ANDNO 16th";
834 ? " OR DOTTED 8th NOTES WITH 2/2
TIME.":? :? "WITH 10 OR 12: NO 16th NO
TES WITH 2/2 TIME.":?
835 ? "WITH 9 OR 12: TRIPLETS ARE ENTE
RED    NORMALLY WITH TEMPOS 9 AND 12 A
ND AS  TWO DOTTED AND ONE NORMAL";
836 TRAP 833:? " NOTE OF THE NEXT FAST
ER SPEED WITH OTHER TEMPOS.  NO TRIPLE
T 16th NOTES WITH 9.":? :INPUT DL
```

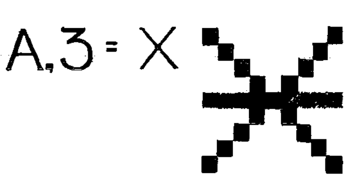
```
837 IF DL<8 OR DL>12 THEN ? CHR$(253):
GOTO 833
838 DL=DL*16
839 IF TS=4 THEN TS=1:DL=DL*0.5
840 DIM N0(V0,R0),N1(V1,R1),N2(V2,R2),
N3(V3,R3),N(11,3),D(3,1)
842 DIM TITLE1$(20),TITLE2$(20),V$(1),
O$(2),N$(2),D$(3),L$(3)
849 ? "}":? :? :? "WHAT IS THE TITLE O
F YOUR SONG?":? :? "CENTER IN 11th SPA
CE":? "WITH FIRST LINE HERE"
850 ? "AND SECOND LINE NEXT.":? "USE U
PPER AND lower":? "CASE & INVERSE vide
o.":INPUT TITLE1$
855 ? :? "CONTINUE TITLE HERE.":INPUT
TITLE2$
910 RESTORE 920:FOR O=0 TO 3:FOR N=0 T
O 11:READ VN:N(N,O)=VN:IF VN=256 THEN
980
920 DATA 243,230,217,204,193,182,173,1
62,153,144,136,128,121,114,108,102,96,
91,85,81,76,72,68,64,60,57,53
930 DATA 50,47,45,42,40,37,35,33,31,29
,0,256
940 NEXT N:NEXT O
980 MVT=44*NV:MN=0:MNT=0:DT=0:SV=0:ST=
0:LN1=101:LNS=39:SN=1
984 FOR R=0 TO 1:FOR C=0 TO 3:D(C,R)=0
:NEXT C:NEXT R
985 ML=FRE(0)
990 FOR V=0 TO V0:FOR R=0 TO R0:N0(V,R
)=0:NEXT R:NEXT V
991 FOR V=0 TO V1:FOR R=0 TO R1:N1(V,R
)=0:NEXT R:NEXT V
992 FOR V=0 TO V2:FOR R=0 TO R2:N2(V,R
)=0:NEXT R:NEXT V
993 FOR V=0 TO V3:FOR R=0 TO R3:N3(V,R
)=0:NEXT R:NEXT V
994 V0=0:V1=0:V2=0:V3=0:V0E=0:CK0=0:CK
1=0:CK2=0:CK3=0:DM=DL*TS:RELN=LN1
995 FOR C=0 TO 3:D(C,0)=0:NEXT C
996 ? "}":? "ENTER TOTAL NUMBER OF COM
BINED          MEASURES.":? :? "HOW MANY
? 1 FOR NO COMBINED MEASURES."
997 ? "               2,3,4 COMBINED MEASUR
ES.":? :? "LIMIT ";NLV;" NOTES PER VOI
CE.":TRAP 996:INPUT CM
998 IF CM<1 OR CM>4 THEN ? CHR$(253):G
OTO 996
1000 DM=DM*CM:MN=MN+CM:MNT=MNT+CM:MV=M
L-FRE(0):ML=FRE(0):MVT=MVT+MV:MVA=MVT/
MN:NM=INT(ML/MVA+0.5)-1
1001 IF H1=0 THEN CK1=1
1002 IF H2=0 THEN CK2=1
1003 IF H3=0 THEN CK3=1
1004 O$="4":O=1:N$="C":N=0:D$="Q":D=DL
/4:L$="MF":L=8
1005 IF CK0=1 AND CK1=1 AND CK2=1 AND
CK3=1 THEN V=0:O=3:N=2:D=0:L=0:GOTO 15
00
1010 ? "}":? :? "          **ENTER MUS
IC**":? :? "THIS IS MEASURE ";MNT:? "N
UMBER OF AVERAGE MEASURES LEFT = ";NM
1015 IF CK0=0 THEN V=0:V$="0":ENT=V0+1
:GOTO 1022
1017 IF CK1=0 THEN V=1:V$="1":ENT=V1+1
:GOTO 1022
1019 IF CK2=0 THEN V=2:V$="2":ENT=V2+1
:GOTO 1022
1021 IF CK3=0 THEN V=3:V$="3":ENT=V3+1
1022 IF ENT=NLV THEN ? CHR$(253):? " T
HIS IS THE LAST NOTE THE COMPUTER    C
AN ENTER IN THIS VOICE!"
1023 IF ENT>NLV THEN ? "}":? " TOO MAN
Y NOTES!":GOTO 5001
1030 ? :? "VOICE ";V$;": ENTRY ";ENT
1031 ? :? "NOTE?":? :? "WHICH? AF,A,AS
":? "          BF,B":? "          C,CS":?
 "        DF,D,DS"
1032 ? "          EF,E,ES":? "           F
,FS":? "          GF,G,GS":? "          R FO
R REST"
1033 ? :? "ENTER  SM FOR SAME NOTE-";N
$;",";D$;",";"OCT";O$;",";L$:INPUT A$
1035 IF A$="R" THEN N=1:O=3:L=0:N$="R"
:L$="OFF":O$="R":? "}":? "VOICE ";V$;"
: ENTRY ";ENT;", REST":GOTO 1121
1038 IF A$="SM" THEN 1211
1040 N$=A$:IF N$="C" THEN N=0:GOTO 112
0
```

```
1045 IF N$="B" THEN N=11:GOTO 1120
1050 IF N$="BF" OR N$="AS" THEN N=10:G
OTO 1120
1055 IF N$="A" THEN N=9:GOTO 1120
1060 IF N$="AF" OR N$="GS" THEN N=8:GO
TO 1120
1065 IF N$="G" THEN N=7:GOTO 1120
1070 IF N$="GF" OR N$="FS" THEN N=6:GO
TO 1120
1075 IF N$="F" THEN N=5:GOTO 1120
1080 IF N$="E" THEN N=4:GOTO 1120
1085 IF N$="EF" OR N$="DS" THEN N=3:GO
TO 1120
1090 IF N$="D" THEN N=2:GOTO 1120
1095 IF N$="DF" OR N$="CS" THEN N=1:GO
TO 1120
1105 ? "}":? CHR$(253):N$="SM":GOTO 10
30
1120 TRAP 1120:? "}":? "VOICE ";V$;":
ENTRY ";ENT;", NOTE ";N$
1121 ? :? "TOTAL DURATION IN MEASURE A
ND PROGRAM ";
1122 ? "   VOICE 0 = ",D(0,0),D(0,1):?
 "   VOICE 1 = ",D(1,0),D(1,1):? "   V
OICE 2 = ",D(2,0),D(2,1)
1123 ? "   VOICE 3 = ",D(3,0),D(3,1)
1124 ? :? "DURATION OF NOTE OR REST?":
? "WHICH? SIXTEENTH":? "        EIGHTH"
1125 ? "        QUARTER":? "        HALF
":? "        WHOLE":? "ADD . FOR DOTTED
  NOTES"
1126 ? "ADD 3 FOR TRIPLET NOTES":? "AD
D T FOR TIED    NOTES"
1127 ? :? "ENTER ED TO EDIT NOTE ";N$:
? "ENTER SM FOR SAME NOTE-";N$;",";D$;
",";"OCT";O$;",";L$:INPUT A$
1129 IF A$="ED" THEN ? "}":GOTO 1030
1131 IF A$="SM" THEN 1211
1137 D$=A$:IF D$(1,1)="S" THEN D=DL/16
:GOTO 1152
1142 IF D$(1,1)="E" THEN D=DL/8:GOTO 1
152
1145 IF D$(1,1)="Q" THEN D=DL/4:GOTO 1
152
```

```
1147 IF D$(1,1)="H" THEN D=DL/2:GOTO 1
152
1150 IF D$(1,1)="W" THEN D=DL:GOTO 115
2
1151 ? CHR$(253):D$="SM":GOTO 1120
1152 IF D$(1,1)=D$ THEN 1158
1154 IF D$(2,2)="T" THEN 1200
1155 TRAP 1156:IF D$(3,3)="T" THEN 120
0
1156 IF D$(2,2)="." OR D$(2,2)="3" THE
N 1158
1157 ? CHR$(253):D$="SM":GOTO 1120
1158 IF N$="R" THEN 1200
1160 ? "}":? :? "VOICE ";V$;": ENTRY "
;ENT;", NOTE ";N$;",";" DURATION ";D$
1161 ? :? "OCTAVE? 4 GOES UP FROM MIDD
LE C":? :? "WHICH? 3,4,5,6"
1162 ? :? "ENTER ED TO EDIT DURATION "
;D$:? "ENTER SM FOR SAME NOTE-";N$;","
;D$;",";"OCT";O$;",";L$:INPUT A$
1163 IF A$="ED" THEN 1120
1164 IF A$="SM" THEN 1200
1166 TRAP 1160:O$=A$:O=VAL(O$):O=O-3:I
F O<0 OR O>3 THEN ? CHR$(253):? "}":GO
TO 1160
1177 ? "}":? :? :? "VOICE ";V$;": ENTR
Y ";ENT;", NOTE ";N$;", DURATION ";D$:
? "OCTAVE ";O$
1178 ? :? "LOUDNESS?":? :? "WHICH? PP,
P,MP":? "        MF,F,FF":? "        OFF
"
1179 ? :? "NORMAL VALUES:":? "  MF FOR
 MELODY"
1180 ? "   P FOR ACCOMPANIMENT":? :? "
ENTER ED TO EDIT OCTAVE ";O$:INPUT A$
1182 IF A$="ED" THEN 1160
1183 L$=A$:IF L$="PP" THEN L=2:GOTO 12
00
1184 IF L$="P" THEN L=4:GOTO 1200
1185 IF L$="MP" THEN L=6:GOTO 1200
1186 IF L$="MF" THEN L=8:GOTO 1200
1187 IF L$="F" THEN L=10:GOTO 1200
1188 IF L$="FF" THEN L=12:GOTO 1200
1189 IF L$="OFF" THEN L=0:GOTO 1200
1192 ? CHR$(253):GOTO 1177
```

```
1200 IF D$(1,1)=D$ THEN 1210
1202 IF D$(2,2)="." THEN D=D*3/2
1204 IF D$(2,2)="T" THEN DT=DT+D:? CHR
$(253):? " ENTER DURATION OF TIED NOTE
!":FOR W=1 TO WT:NEXT W:GOTO 1120
1206 TRAP 1208:IF D$(3,3)="T" THEN DT=
DT+D:? CHR$(253):? " ENTER DURATION OF
 TIED NOTE!":FOR W=1 TO WT:NEXT W:GOTO
 1120
1208 IF D$(2,2)="3" THEN D=D*2/3
1210 D=D+DT:DT=0:IF D<>INT(D) OR D<8 T
HEN 1940
1211 IF D=3 AND N<>1 AND N<>0 THEN ? C
HR$(253):? " NOTE TOO HIGH!":FOR W=1 T
O WT:NEXT W:GOTO 1160
1212 D(V,0)=D(V,0)+D:D(V,1)=D(V,1)+D
1214 IF D(V,0)>DM THEN D(V,0)=D(V,0)-D
:D(V,1)=D(V,1)-D:GOSUB 1940
1216 TRAP 5001:IF V=0 AND D(0,0)=DM TH
EN CK0=1:? CHR$(253)
1218 IF V=1 AND D(1,0)=DM THEN CK1=1:?
 CHR$(253)
1220 IF V=2 AND D(2,0)=DM THEN CK2=1:?
 CHR$(253)
1222 IF V=3 AND D(3,0)=DM THEN CK3=1:?
 CHR$(253)
1224 IF V=0 THEN 1500
1226 IF V=1 THEN 1600
1228 IF V=2 THEN 1700
1230 IF V=3 THEN 1800
1500 NO(VO,1)=N(N,0):NO(VO,2)=L:NO(VO,
3)=D:IF N(N,0)=256 THEN GOTO 700
1510 VO=VO+1:VOE=VO:GOTO 1005
1600 N1(V1,1)=N(N,0):N1(V1,2)=L:N1(V1,
3)=D:V1=V1+1:V1E=V1:GOTO 1005
1700 N2(V2,1)=N(N,0):N2(V2,2)=L:N2(V2,
3)=D:V2=V2+1:V2E=V2:GOTO 1005
1800 N3(V3,1)=N(N,0):N3(V3,2)=L:N3(V3,
3)=D:V3=V3+1:V3E=V3:GOTO 1005
1901 ? :? CHR$(253);" VOICE UNAVAILABL
E!":FOR W=1 TO WT:NEXT W:GOTO 5000
1940 ? :? CHR$(253);" DURATION UNAVAIL
ABLE!":FOR W=1 TO WT:NEXT W:D$="Q":D=D
L/4:GOTO 1120
5000 TRAP 5000:? "}":? "    **EDIT MEA
SURE OF MUSIC**":? :? "VOICE?":? :? "W
HICH? 0,1,2,3":INPUT V
5001 VO=0:V1=0:V2=0:V3=0
5002 IF V<>0 THEN VO=VOE
5004 IF V=1 AND H1=0 THEN 1901
5005 IF V<>1 AND H1=1 THEN V1=V1E
5006 IF V=2 AND H2=0 THEN 1901
5007 IF V<>2 AND H2=1 THEN V2=V2E
5008 IF V=3 AND H3=0 THEN 1901
5009 IF V<>3 AND H3=1 THEN V3=V3E
5010 IF V<0 OR V>3 THEN ? CHR$(253):GO
TO 5000
5016 IF V=0 THEN D(0,1)=D(0,1)-D(0,0):
D(0,0)=0:CK0=0
5017 IF V=1 THEN D(1,1)=D(1,1)-D(1,0):
D(1,0)=0:CK1=0
5018 IF V=2 THEN D(2,1)=D(2,1)-D(2,0):
D(2,0)=0:CK2=0
5019 IF V=3 THEN D(3,1)=D(3,1)-D(3,0):
D(3,0)=0:CK3=0
5020 ? :? CHR$(253);"EDIT VOICE ";V:FO
R W=1 TO WT:NEXT W:GOTO 1004
6500 IF ST=1 THEN 7510
6502 ? "}":? "           **SAVE MUSIC**
":? :? "MOVE CURSOR TO LINE NUMBER (LN
) AND"
6505 ? "PRESS RETURN TO ENTER INTO PRO
GRAM.   THEN TYPE CONT AND PRESS RETUR
N TO    CONTINUE."
6508 ? :? "LN"
6510 ? "40 ?#6:?#6; ";CHR$(34);TITLE1$
;CHR$(34);" :?#6:?#6; ";CHR$(34);TITLE
2$;CHR$(34)
6511 IF H1=0 THEN ? "60"
6512 IF H2=0 THEN ? "70"
6513 IF H3=0 THEN ? "80"
6514 STOP
7510 ST=1:ORD=0:VO=0:V1=0:V2=0:V3=0:LN
1=LN1-1
7520 ? :? "}":? "LN "
7522 FOR LN=1 TO 6:LN1=LN1+1
7525 ? LN1;" DATA ";
7530 FOR NS=1 TO 9
7532 IF NS=9 AND NO(VO,0)=ORD AND NO(V
0,1)=256 THEN 7540
```

```
7535 IF NS=9 THEN ? CHR$(126):GOTO 757
5
7540 IF NO(VO,0)=ORD AND NO(VO,1)=256
THEN ? CHR$(126):? LN1+1;" DATA ";NO(V
0,1);",";NO(VO,2);",";NO(VO,3):GOTO 75
80
7545 IF NO(VO,0)=ORD THEN ? NO(VO,1);"
,";NO(VO,2);",";NO(VO,3);",";:VO=VO+1:
ORD=ORD+1:NEXT NS
7550 IF N1(V1,0)=ORD THEN ? N1(V1,1);"
,";N1(V1,2);",";N1(V1,3);",";:V1=V1+1:
ORD=ORD+1:NEXT NS
7560 IF N2(V2,0)=ORD THEN ? N2(V2,1);"
,";N2(V2,2);",";N2(V2,3);",";:V2=V2+1:
ORD=ORD+1:NEXT NS
7570 IF N3(V3,0)=ORD THEN ? N3(V3,1);"
,";N3(V3,2);",";N3(V3,3);",";:V3=V3+1:
ORD=ORD+1:NEXT NS
7575 NEXT LN
7580 ? :? "ENTER LINE NUMBERS AND CONT
INUE.":STOP
7590 IF NO(VO,1)<>256 THEN GOTO 7520
7600 LN1=LN1+1:GOTO 30
8450 ? "}":? "        **SAVE ON DISKETT
E**":? :? "SAVE PLAY PROGRAM AND DATA
WITH THE"
8451 ? "INITIAL GROUP OF MEASURES BUT
ONLY    DATA IN SUBSEQUENT SAVES."
8452 ? :? "INSERT BLANK DISKETTE!":? :
? "ENTER    LINE NUMBER AND CONTINUE."
:?
8453 ? " LN ":? "8460 LIST ";CHR$(34);
"D:SONG";SN;".ENT";CHR$(34);",";LNS;",
";LN1:STOP
8454 ? "}":? :? "PROGRAM BEING SAVED."
8461 IF LNS=39 THEN LNS=101:? "}":? :?
 "ENTER LINE NUMBERS TO CLEAR MEMORY F
ORNEXT GROUP OF MEASURES."
8462 FOR W=1 TO WT:NEXT W:LN=LNS-1
8463 ? "}":? "LN ":LNF=LNS+15:IF LNF>=
LN1 THEN LNF=LN1
8464 FOR W=LNS TO LNF:LN=LN+1:? LN:NEX
T W
8465 SN=SN+1:? :? "ENTER LINE NUMBERS
AND CONTINUE.":STOP
8466 LNS=LNF+1:IF LNS<=LN1 THEN 8463
8468 ? "}":? :? "LOAD OR ADD THIS MUSI
C PROGRAM AND"
8469 ? "ADDITIONAL DATA TO ANOTHER PRO
GRAM    WITH ENTER "D:SONG#.ENT""
8470 LNS=LN1:FOR W=1 TO WT:NEXT W:GOTO
 985
9031 TRAP 30:? :? "SAVE ON CASSETTE OR
":? :? "RE-ENTER LINE NUMBERS":? :? "W
HICH?":? :? "ENTER 1 TO PLAY"
9450 ? "}":? "        **SAVE ON CASSETT
E**":? :? "SAVE PLAY PROGRAM AND DATA
WITH THE"
9452 ? :? "ENTER LINE NUMBER  AND CONT
INUE.":?
9453 ? " LN ":? "8460 LIST ";CHR$(34);
"C";CHR$(34);",";LNS;",";LN1:STOP
9454 ? "}":? :? "ON BELL, SET RECORDER
 TO RECORD AND   PRESS RETURN."
9469 ? "ADDITIONAL DATA TO ANOTHER PRO
GRAM    WITH ENTER "C""
```



Character Generator

by Alan J. Zett

***Atari Character Generator* is an Atari utility program requiring 16K RAM.**

The first thing to do before running the *Character Generator* is to press SYSTEM RESET. Then load the program, and type RUN. It is absolutely necessary to press SYSTEM RESET prior to running any program using a modified character set. Forgetting to do so can cause the computer to lock up and lose the program currently in memory.

After about twenty seconds you should see all the characters (except inverse) displayed on the left side of the screen; a large box displayed on the right side; and at the bottom the message "Edit character or Save file?." Let's assume for now that you type "E."

The message "X,Y coordinate of the character?" will now be displayed. This is prompting you for the horizontal (X) position of the character, which is between 0 and F, and the vertical (Y) position, which is between 0 and 7. Type them in together. For this example we'll type 0,2 (the comma is optional) which corresponds to the blank space character.

The computer will now put that character into a small box, and an expanded version in the large box next to it. In this case the character is blank, so nothing is put in either box.

The message at the bottom shows you the nine different commands available to you in the modify mode. You should also see a blinking "+" in the large box on the right. This is your cursor. It shows you where the next modification will take place. Note: Only DELETE and RETURN make actual changes to the character set as stored in memory; the other commands affect only the display.

To plot a block in the character, use the SET command. Use the arrow keys to position the cursor where you want to make a change, and press "S" to turn the block on. To do the opposite, type "R" to RESET the block (make it blank again), positioning the cursor in the same way as for the SET command. If you want to erase the entire drawing pad, type "E" for ERASE. Or, to start with all blocks on, type "W" to WHITE out all blocks. Reversing the drawing is done by typing "I" for INVERSE.

If you want to load in another character with which to start your design, type "C" for COPY. You will be asked to select a character, and its pattern will then be copied onto the drawing pad.

If you accidentally choose the wrong character to modify, you can cancel the MODIFY mode at any tme by hitting the ESC key. This allows you to ESCAPE to the "X,Y of character" mode. The DELETE command lets you change your mind about a modification by restoring the current character to its unmodified form.

Finally, when everything is correct, hit RETURN to make the change both in memory and on the screen. Remember that this change is not permanent, and can be changed again by using any of the commands.

For our example, let's draw a series of vertical stripes in the large box. When everything is just right, press RETURN. Right in front of your eyes, all blank spaces on the screen should turn a different color.

You can keep on modifying all night if you want, but for now, we'll call it a day. When you see the message "X,Y of ...," press RETURN again. This will take you back to the original menu. From here, select the "S" option to save the file. The computer will ask, "Want to SAVE (Y/N) ?." In case you hit "S" by accident, type N. Otherwise, type Y and hit RETURN. The prompt, "To Cassette or Disk?" will be displayed; answer with a "C" or "D."

If you said "D," then answer the next question with the 8-letter filename plus a 3-letter extension, all preceded by a "D:". Answer the next question "Is this right?" with a "Y" or "N", and hit RETURN.

If you said "C", for a cassette save, then you'll hear two beeps. Prepare the cassette and press RETURN. After the save, type "Y" to continue or "N" to stop.

Now that you've saved it, what can you do? This is the best part! Write your program between lines 2 and 31999 using the characters that will be modified where necessary. When it's all done and totally debugged, type either ENTER"C:" for cassette or ENTER"D:filename.ext" for disk. This will merge the custom character set into your program, and it should run perfectly the first time.

Two final notes: First, the new program will take an extra 1.25K of memory and will take longer to initialize. Second, after every GRAPHICS command you must insert a POKE 756, PEEK (106)+1. This is because the GRAPHICS command restores the character set pointer in memory. The new set is still there, but the pointer must be reset to access it. One benefit from this is that you can toggle back and forth between the two character sets by typing POKE 756, 224 to get the original set and POKE 756, PEEK (106)+1 for the new set. Note that PRINT CHR$(125) clears the screen as does GRAPHICS 0, but does not affect the current character set.

Editor's note:

XFR$, which you will find printed as a series of periods in the program listing, actually contains a Machine Language routine that quickly sets up a redefined character set for the Atari. It is very difficult to represent this string in the listing of the program, so here are explicit instructions on how to type it.

In these instructions, each key is represented by something between brackets. Thus [h] means to type a lower-case "h"; likewise, [Atari] means to press the Atari logo key, which is found next to the right-hand shift key, and [CTRL-N] means to press the "N" key while holding down the "CTRL" key. Special note: [1] is a the numeral one — it is not the lower-case version of "L"!

Type the following sequence exactly to produce XFR$:
[h] [Atari] [)] [Atari] [CTRL-,]
[Atari] [CTRL-E] [K] [CTRL-E] [M]
[)] [CTRL-.] [CTRL-E] [N] [%]
[Atari] [j] [CTRL-X] [i] [CTRL-A]
[Atari] [CTRL-E] [L] [spacebar]
[Atari] [CTRL-,] [Atari] [1] [M]
[CTRL-Q] [K] [H] [P] [y] [f] [L] [f]
[N] [%] [N] [I] [d] [P] [m] [Atari]
[CTRL-.]

Variables

A$: Misc. string input.
B: Binary bit value counter.
B(i): Modified character pointer array (-1 = not modified).
C: Character byte value.
CC: Current character at cursor position.
CH: Command character.
CU: Cursor character.
CX: Cursor X position.

CY: Cursor Y position.
FL: Modify flag.
HEX$: Used in converting hex input to decimal.

KP: Key pressed? (No = 255.)
START: Starting location in memory of new character set.
W, X, Y, Z: Misc.

```
   SS SS SS SS SS SS SS SS SS SS SS
   SS                            SS
   SS         Atari  BASIC       SS
   SS   'Character  Generator'   SS
   SS     Author: Alan J. Zett   SS
   SS      Copyright (c) 1982    SS
   SS SoftSide Publications, Inc SS
   SS                            SS
   SS SS SS SS SS SS SS SS SS SS SS
```

Initialization.

```
10 OPEN #1,4,0,"K":OPEN #2,12,0,"S"
12 CLR :DIM B(127),HEX$(16)
14 DIM A$(14):GOSUB 10000
20 HEX$="0123456789ABCDEF"
22 SETCOLOR 2,9,2:SETCOLOR 4,4,4
24 POKE 82,21
```

Redraw entire display.

```
30 ? CHR$(125):POKE 752,1
32 POSITION 19,0
33 ? " CHARACTER GENERATOR "
34 POSITION 19,1
35 ? "   BY ALAN J. ZETT    "
36 FOR X=7 TO 16 STEP 9:POSITION 2,X
38 ? " 0123456789ABCDEF ";:NEXT X
40 FOR X=8 TO 15:POSITION 2,X
42 ? CHR$(X+168);:POSITION 19,X
44 ? CHR$(X+168);:NEXT X
50 POSITION 4,5:? "Character set:";
60 POKE 766,1:FOR X=0 TO 7
62 FOR Y=0 TO 15:POSITION Y+3,X+8
64 ? CHR$(X*16+Y);:NEXT Y:NEXT X
66 POKE 766,0:POKE 752,1
70 POSITION 26,7:? "qrrrrrrre";
71 REM <CTRL> 'QRRRRRRRE'
72 POSITION 26,16:? "xrrrrrrrc";
73 REM <CTRL> 'XRRRRRRRC'
80 FOR X=8 TO 15:POSITION 26,X
82 ? "=        =";:NEXT X
83 REM <SHIFT> '=        ='
90 POSITION 24,14:? "qrd";
91 REM <CTRL> 'QRD'
92 POSITION 24,15:? "= =";
93 REM <SHIFT> '= ='
94 POSITION 24,16:? "zrx";
95 REM <CTRL> 'ZRX'
```

Select main program options.

```
100 GOSUB 9000:POSITION 2,20
102 POKE 752,0
104 ? "Edit character or Save file";
106 INPUT A$
108 IF A$(1,1)="S" THEN 570
110 IF A$(1,1)<>"E" THEN 100
```

Input and adjust value of character.

```
120 GOSUB 9000:POSITION 2,20:TRAP 66
122 ? "X,Y coordinate of the character
";:INPUT A$
130 IF A$(LEN(A$))<"0" OR A$(LEN(A$))>
"7" THEN 120
140 TRAP 33333:Y=VAL(A$(LEN(A$))):X=1
150 IF A$(1,1)<>HEX$(X,X) THEN X=X+1:I
F X<17 THEN 150
160 X=X-1:IF X>15 THEN 120
170 POKE 752,1:Z=X+Y*16
172 POSITION 25,15:POKE 766,1
174 ? CHR$(Z);:POKE 766,0
180 IF Z<32 THEN Z=Z+64:GOTO 200
190 IF Z<96 THEN Z=Z-32
```

Draw character onto pad.

```
200 POKE 82,27
210 FOR Y=0 TO 7:X=0:C=PEEK(START+Y+Z*
8):B=256
220 B=B/2:IF B<1 THEN 270
```

```
230 POSITION X+27,Y+8:X=X+1
240 IF (C-B)<0 THEN ? " ";:GOTO 260
250 IF (C-B)>=0 THEN ? "_";:C=C-B
260 GOTO 220
270 NEXT Y
```

Get command; branch to appropriate routine.

```
280 GOSUB 9000:POSITION 2,20:POKE 82,2
290 POKE 752,1:? "Copy, Delete, Erase,
 Inverse, Reset,":? :? "Set, White, ES
Cape, RETURN to store.";
300 POKE 752,1:POKE 764,255:CX=27:CY=8
310 POSITION CX,CY:GET #2,CC
320 POSITION CX,CY:PUT #2,CC:KP=PEEK(7
64):CU=43:IF CC<>32 THEN CU=171
330 IF KP=255 THEN POSITION CX,CY:PUT
#2,CU:GOTO 320
340 GET #1,CH:IF CH=155 THEN 470
350 IF CH=27 THEN 120
352 IF CH=ASC("E") THEN 790
354 IF CH=ASC("C") THEN 820
356 IF CH=ASC("D") THEN 810
357 IF CH=ASC("W") THEN 1000
358 IF CH=ASC("I") THEN 1020
360 IF CH=ASC("-") THEN CY=CY-1
370 IF CH=ASC("=") THEN CY=CY+1
380 IF CH=ASC("+") THEN CX=CX-1
390 IF CH=ASC("*") THEN CX=CX+1
400 IF CH=ASC("S") THEN CC=160:GOTO 32
0
410 IF CH=ASC("R") THEN CC=32:GOTO 320
```

Move cursor on pad.

```
420 IF CX<27 THEN CX=34
430 IF CX>34 THEN CX=27
440 IF CY<8 THEN CY=15
450 IF CY>15 THEN CY=8
460 GOTO 310
470 POSITION CX,CY:PUT #2,CC
```

Modify character.

```
480 FOR Y=0 TO 7:X=0:B=256:CC=X:C=X
490 B=B/2:IF B<1 THEN 540
500 POSITION X+27,Y+8:X=X+1:GET #2,CC
510 IF CC=32 THEN 530
520 IF CC=160 THEN C=C+B
530 GOTO 490
540 POKE START+Y+Z*8,C
550 NEXT Y:POSITION 34,15
560 PUT #2,CC:B(Z)=Z*8
562 FL=0:FOR Y=0 TO 7
564 IF PEEK(START+Y+Z*8)<>PEEK(57344+Y
+Z*8) THEN FL=1
566 NEXT Y:IF FL=0 THEN B(Z)=-1
568 GOTO 120
```

Save file. Note: Correct spacing is critical in these lines.

```
570 PRINT CHR$(125):POKE 82,2
580 TRAP 990:POSITION 2,5
582 ? "You want to SAVE (Y/N)";
584 INPUT A$:IF A$(1,1)<>"Y" THEN 30
590 ? "To Cassette or Disk";:INPUT A$
600 IF A$<>"D" THEN 620
610 ? "Enter file 'D:FILENAME.EXT'"
612 INPUT A$:? "Is '";A$;"' correct ?"
;
614 GET #1,CH:IF CH=78 THEN 610
620 OPEN #3,8,0,A$
630 PRINT #3;"1 GOSUB 32000:CLR"
640 PRINT #3;"32000 POKE 106,PEEK(106)
-5:GRAPHICS 0:START=(PEEK(106)+1)*256:
POKE 756,START/256:POKE 752,1"
650 PRINT #3;"32010 DIM XFR$(38):XFR$=
";CHR$(34);"........................
............";CHR$(34)
651 REM Line 650: Replace string of
.'s with the string in the documen-
tation.
660 PRINT #3;"32020 Z=USR(ADR(XFR$)):R
ESTORE 32100"
670 PRINT #3;"32030 READ X:IF X=-1 THE
N RESTORE :RETURN"
680 PRINT #3;"32040 FOR Y=0 TO 7:READ
Z:POKE X+Y+START,Z:NEXT Y:GOTO 32030"
690 LN=32100:LI=1:FOR Z=0 TO 127
700 IF B(Z)=-1 THEN 750
710 PRINT #3;STR$(LN);" DATA ";
```

```
720 PRINT #3;STR$(B(Z));
730 FOR Y=0 TO 7:PRINT #3;",";STR$(PEE
K(START+Y+Z*8));:NEXT Y
740 PRINT #3;"":LN=LN+LI
750 NEXT Z:PRINT #3;STR$(LN);" DATA ";
STR$(-1):CLOSE #3:TRAP 33333
760 POKE 764,255:? :? "WANT TO CONTINU
E";:INPUT A$
770 IF A$(1,1)="Y" THEN 30
780 ? CHR$(125):END
```

Erase command.

```
790 FOR Y=8 TO 15:POSITION 27,Y
800 ? "        ";:NEXT Y:GOTO 300
```

Delete command.

```
810 B(Z)=-1:FOR Y=0 TO 7:POKE START+Y+
Z*8,PEEK(57344+Y+Z*8):NEXT Y:GOTO 120
```

Copy command.

```
820 GOSUB 9000:POSITION 2,20
822 ? "X,Y of character to copy";
824 INPUT A$
830 IF A$(LEN(A$))<"0" OR A$(LEN(A$))>
"7" THEN 820
840 POKE 752,1:Y=VAL(A$(LEN(A$))):X=1
850 IF A$(1,1)<>HEX$(X,X) THEN X=X+1:I
F X<17 THEN 850
860 X=X-1:IF X>15 THEN 820
870 W=X+Y*16
880 IF W<32 THEN W=W+64:GOTO 900
890 IF W<96 THEN W=W-32
900 FOR Y=0 TO 7
910 X=0:C=PEEK(START+Y+W*8):B=256
920 B=B/2:IF B<1 THEN 970
930 POSITION X+27,Y+8:X=X+1
940 IF (C-B)<0 THEN ? " ";:GOTO 960
950 IF (C-B)>=0 THEN ? "_";:C=C-B
960 GOTO 920
970 NEXT Y
980 GOTO 280
```

Error trap.

```
990 ? :? :? " INPUT ERROR, RE-ENTER. "
:GOTO 582
```

White command.

```
1000 FOR Y=8 TO 15:POSITION 27,Y
1010 ? "________";:NEXT Y:GOTO 300
```

Inverse command.

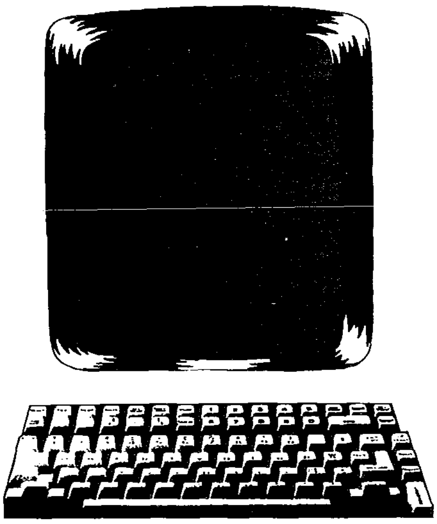
```
1020 FOR Y=8 TO 15:FOR X=27 TO 34
1030 POSITION X,Y:GET #2,CH:POSITION X
,Y
1040 IF CH=32 THEN PUT #2,160
1050 IF CH=160 THEN PUT #2,32
1060 NEXT X:NEXT Y:GOTO 300
```

Erase old messages.

```
9000 POKE 752,1:POSITION 0,20
9010 ? "!!!!!";:POKE 752,0:RETURN :REM
 Replace ! with [ESC][SHIFT][DELETE]
9020 REM 5 <ESC><SHIFT><DELETE>'S
```

Download new character set, and reset modification pointers to -1.

```
10000 POKE 106,PEEK(106)-5:GRAPHICS 0
10010 START=(PEEK(106)+1)*256
10020 POKE 756,START/256:POKE 752,1
10030 ? "INITIALIZING . . ."
10040 DIM XFR$(38)
10050 XFR$="..........................
............"
10051 REM Line 10050: Replace .'s with
the string listed in the documenta-
tion.
10060 Z=USR(ADR(XFR$)):FOR Z=0 TO 127
10070 B(Z)=-1:NEXT Z:RETURN
32767 SAVE "D:CHARGEN.AJZ":STOP
```



RANDOM ACCESS DATABASE

by Mark Pelczarski
Atari version by Paul Marentette

***Random Access Database* is an information management program for an Atari 400/800, requiring 32K RAM and a disk drive running under DOS 2.**

In-memory databases are limited in size because data must fit into available RAM. For example, if you had a 40K machine, you could have a database of approximately 170 90-byte records. With the random access database (and 480 free sectors on a disk) more than 770 of these records could fit in the database. Moreover, in-memory database managers must save all data to disk whenever the data changes. With a large quantity of data, this is both time-consuming and inefficient.

Before examining the *Random Access Database,* let's consider how Atari handles random disk access. The NOTE and POINT instructions are used to keep track of where each record resides on the disk. When a file is opened in mode 12 (OPEN #IOCB, 12,0,"D:FILENAME") an internal pointer positions the disk's read/write head at the beginning of the file. Under program control, the POINT #IOCB, Sector,Byte instruction is then used to reposition the disk head at any sector and byte in the file.

In this manner the computer can automatically calculate where a record resides on the disk, and bring only that record into memory. Only one complete record ever resides in RAM; this frees up the bulk of memory for storing the pointers (disk addresses) for each record in the database. Since each pointer uses only 3 bytes, a much larger database can be created.

The pointers, consisting of the sector (1-720) and byte (1-125) address of each record, are stored in P$. They could be stored in a numeric array, but that would require 12 bytes of memory for each sector-byte pair. Instead, by using the CHR$ and ASC functions, the two numbers can be stored in 3 bytes. The first two bytes store a sector number. Byte #1 contains the number of 256's in the sector number; byte #2 contains the remainder; and byte #3 contains the byte portion of the disk address. Thus an address of sector 550, byte 63 would have a pointer in the P$ string consisting of CHR$(2), CHR$ (38), and CHR$(63).

The routine from line 1070 to 1110 writes blank records into the disk file and stores their NOTEd addresses in the P$ string using the CHR$ function. Lines 170 to 180 contain a reverse process which uses the ASC function to decode the P$ characters into byte and sector numbers and then point the disk head to the calculated address. The program then reads (line 1160) or writes (line 1140) a record at that address.

Stepping Through the Program

Lines 110 to 700 contain variable initialization, frequently used subroutines, and the main menu and disk directory read routines.

Now let's examine the file initializing routine. Starting at line 850 the number and names of headings are established. The numeric array B(NH + 1) is used to store the starting bytes of each field since we are allowing fields to have different lengths.

Throughout the program the, S, E and L variables specify the start, end, and length of a field within the total record. These values are recalculated any time the field in question changes.

Once the headings are known, the program calculates the maximum number of records that can be accommodated, given the two limiting factors of available RAM and disk space. Then the tricky business starts. We could set aside enough disk space for MX records; but, if the database is a small one, disk space will be unnecessarily dedicated to the database file. We could add a new record to the end of the file each time one is needed; but that is wasteful too — Atari's File Management System grabs a new disk sector every time a record is added (using IO mode 9) and ignores any vacant bytes that might have existed on previously used sectors.

The way I solved this problem was to "block" records in groups that come as close as possible to filling complete sectors. Up to 50 records (an arbitrary number) can be in a block. An efficient block size between 1 and approximately 50 is calculated in lines 1000 to 1040, and then the subroutine at line 1070 is called to do the actual setting up of file space. The variable RA keeps track of total file space already set aside. When more space is needed, another block is added on.

Every database file you create will have two disk files associated with it: The one with a ".DAT" extension will hold the actual records, and the other with a ".HDG" extension will hold the heading names, file size information, and the P$ string containing the pointers to all records in the ".DAT" file. The heading file is saved in lines 1180 to 1260. Variables BLK and RA are saved along with the others so that the database program always knows how a particular file is blocked and what space is already set aside. The ".HDG" file will grow in size as the database grows; as more records are added, the pointer string gets longer, and that string is saved up to its current length.

When an old file is loaded (lines 720 to 800), the ".HDG" file is read first and variables are dimensioned as needed. Then a routine at line 820 is called to read into memory one complete set of fields for the first heading. In other words, the first heading's contents for every record are placed in memory. The CH variable keeps track of which heading is the current one. If a sort or search is done on a different field, that field's contents for every record are read in using this same routine.

The add and change records routines share a common input data subroutine (lines 1710-1730). If, when a record is being added (line 1600), available file space (RA) has been filled, a new block of space is created by calling the routine at line 1070. The variable IO is set to 9 so that the file is opened in the append mode. In the change record section the variable CS is checked to see if any changes were made to the current record. If so (CS = 0), then the subroutine at line 1140 is called to rewrite the record at the same location on the disk from which it was read.

In the delete and sort records

routines the real beauty of random access is seen in action. When a record is deleted from the database it is not removed from the disk. Instead, the pointer to that record's disk space is moved to the end of the P$ pointer string (line 1900) so that the deleted record's disk space can be made available for new records. In the sort routine, whenever the sort field is moved around, the corresponding pointers in the P$ string are moved around too. In effect, the records are never moved on the disk — only the pointers get sorted!

Data is automatically saved when you quit the program. (Don't exit the program any other way than the Q menu option!) If you want to save a copy of your file in the midst of working, the save option will save a copy of your current heading and pointer file. The data itself is always saved automatically, immediately after it is entered or changed.

Running The Database

The first choice you will have is whether to initialize a new file or load an existing file. The first time you use it, you'll have to initialize a file. Thereafter, when you want to access that data, you will load the file. Anytime you want to create a new file with a different type of data, you'll use the initialize option. Several different files will fit on a disk and you can use as many different data disks as you like. A few examples of files would be a mailing list (name, address, city, etc.), checkbook list (to whom, withdrawals, deposits...), and an inventory list (stock number, description number, in stock, on order, etc.). Whatever records you want to keep can usually be stored in this type of database format.

To initialize a new file, you must give your file a name, then tell the computer how many headings you want and the names and lengths of the headings. An example would be a file named "Addresses," with six headings: Name, Street Address, City, State, Zip Code, and Phone Number. You might want to add an extra heading (or more) for some kind of code. I might use "Computer" for my seventh heading, so I would know what kind of computer a particular person owns.

Your data will be organized into what you can picture as a table. The headings should be your column headings, and each row would have one set of information across those headings. A set of such information is called one record. Once a file has been created, any future time that you use the database you only have to give the file name ("Addresses" is our example) and all the information will automatically be loaded from disk.

The Main Options

After initializing or loading, you are given a list of choices for manipulating your database. Here are the choices in brief form:

(S) SAVE current data
(P) PRINT data (to screen or printer)
(A) ADD new data
(C) CHANGE some of your data (such as an address change)
(D) DELETE a record
(T) SORT
(F) FILE names - catalogs the disk
(N) NEW data file - equivalent to quitting and re-running program)
(Q) QUIT - done (don't use any other method to get out or you may lose data)

Adding A Record

This is your logical first choice since, with no data in memory, the other options aren't too useful. Choose (A) from the options page and you'll be asked for information to fill each of your headings for one record. After you've filled one record, you'll be returned to the options page. Because of disk limitations, commas, colons, and semicolons don't work in the data.

Also see the note below about searching and sorting numeric fields.

Printing A Record

To see if your data is really there, type "P" to print your record. The program will ask if you want it put in a special format (S) or default format (D). Choose (D) for the moment. After choosing, you'll be asked if you want it on the screen (S) or the printer (P). Then, after that choice, a list of headings will be displayed, followed by the choices "Begin" and "Return to Menu." Type the number next to the word "Begin," and press RETURN. Each record you have in memory will be displayed in sequence. If you're printing them to the screen, pressing any key advances to the next record. The ESC key returns you to the option page. All the other choices mentioned above will be explained under "searching" and "formatting."

Searching

When printing, changing, or deleting records, you have the choice of selecting individual items, subsets of your data, or the entire set of data. This is done through the search routine. When you used the print routine above, you chose to print all the data by selecting "Begin" before any other choice. Each of the headings is also listed at that point, along with "Record Number." By choosing the number beside any of the headings or "record number," you elect to do a search under that heading. You are then asked if you want to look for an item that is less than or equal to, equal to, or greater than or equal to a value you'll give. After choosing 1, 2, or 3, respectively, you'll be asked for a value for comparison. Example: If you want to search for all records with names starting with A through G, you would want NAME,choice 1,G, where "G" is the value used for comparison. If you want all records from number 20 through the end of the file, you would choose RECORD NUMBER,choice 3,20.

You also have the option of specifying the beginning of a value for comparison. If you wanted all records from people whose ZIP code starts with a "60" (as 60185), you can specify ZIP CODE,choice 2,60*. The asterisk says that anything may follow. This is also an easy way to find records without knowing exact information. If you can't spell Pelczarski, you can try "PEL*" and you'll find the record.

To start the actual search, you must choose "Begin." A hidden option here is that you may specify several search criteria (up to 8). I might, for example, want to find everyone in my list whose ZIP code begins with "60" and who owns an Atari. I would specify ZIP CODE,equal to,60*; then I'd specify COMPUTER,equal to,ATARI. Then I would choose "Begin." Having done this, I'd be asked if I want the item to meet *all* of the conditions or *any* of them. If I choose "all," I would find only those people who both own an Atari and have a ZIP code beginning with "60." "Any" would give me all Atari owners plus all the people in ZIP code "60."

Changing Records

To change a record, choose (C) from the options menu. After you specify the search criteria you want, if any, the appropriate records will appear on the screen. The items under each heading will then be shown in sequence, and the program will wait for you to strike "K" to keep, "C" to change, or "R" to keep the remainder of the record. If you strike "C," you'll be asked to provide the new information.

Deleting Records

After choosing (D) from the options menu, and going through the search steps, the records in question will be displayed, and you'll have to verify that you want to delete the particular record, because, once deleted, a record is gone forever. Strike the "Y" key to delete.

Saving a File

When you want to sign off for the day, just choose the (Q) option; your file is automatically saved, and the program will end. The (S) option will save your current file, but will not end the program. It is a good idea to save important information on two separate disks, using one as a backup.

Sorting

The (T) option of the menu allows your items to be sorted in ascending or descending order, under any heading. Alphabetic items are sorted alphabetically, and numeric items are sorted as strings. This means that numbers won't always come out the way you want. 125, 34, and 7 will come out in that order because strings are sorted according to their first character. To get a correct numeric sort, you must add leading zeroes to the maximum number of places, so that all the numbers in your file are the same length. In our example, you should use 007, 034, and 125; these values will sort properly.

Disk Directory

Menu option (F) will show you the filenames of the files on your disk.

Switching Data Files

You can load or create a new file without rerunning the program by selecting menu option (N). Be sure to verify that your current file has been saved.

Formatting Output

This program's formatter may well be the most versatile around. Although it can't do everything, it does much that "professional" database managers cannot. Basically, you can specify the exact form in which you want each record printed. Each record is printed in sequence, meaning that you can't mix records across a page. Another limitation is that there is no way to print a one-time heading.

But let's move on to what the formatter *can* do. You can specify which headings shall be printed, and where, as well as additional character strings that are not stored as part of your database file (examples of these might be your company name or an expanded version of a heading, rather than the heading itself).

To create a format, choose the special-format option when printing. You'll be asked if you want to load or create one. Naturally, the first time, you'll have to create it. Draw out exactly what you want printed for your form. You'll be telling the computer, line by line, what it looks like. Your choices are (1) Heading, (2) Item, (3) Tab, (4) Next line, (5) String, and (6) End. Here's one example using the "ADDRESSES" file I mentioned earlier. The format will print mailing labels like this:

```
Mark Pelczarski
1206 Kings Circle
West Chicago IL
                60185
```

Here are the format commands (numerically, my headings are 1 Name, 2 Address, 3 City, 4 State, 5 ZIP, 6 Phone, 7 Computer):

Commands	What To Type
Item, Name	2, 1
Next Line, 1	4, 1
Item, Address	2, 2
Next Line, 1	4, 1
Item, City	2, 3
Tab, 16	3, 16
Item, State	2, 4
Next Line, 1	4, 1
Tab, 12	3, 12
Item, ZIP	2, 5
Next Line, 3	4, 3
End	6

The "1" after the next line means to skip down one line. The "3" at the end skips down three lines before printing the next label. Note that none of the actual headings is used in this format, nor is the phone number.

Another example is a format that will print a separate little form for each person in the database. This is what I'll have printed:

```
----------
THIS PERSON HAS AN

                    ATARI

NAME JOE TATE
PHONE 555-1212
----------
```

Here's the format to do it:

String, ----------
Next Line, 1
String, THIS PERSON HAS AN
Next Line, 2
Tab, 9
Item, Computer
Next Line, 2
Heading, Name (Hdg. #1)
Tab, 7
Item, Name (Item #1)
Next Line, 1
Heading, Phone (Hdg. }6)
Tab, 7
Item, Phone
End

I'll let the top line of the next item to be printed be the bottom line for the previous item, so I can just end the format after printing the last item.

That's all there is to formatting. Play around with it a bit to see what it does for you. After a format is created, you'll be asked to name it; it will automatically be saved to disk. Later, you'll be able to load it by name when you need it.

```
  SS SS SS SS SS SS SS SS SS SS SS
  SS                             SS
  SS         Atari  BASIC        SS
  SS        'Database  II'       SS
  SS           Authors:          SS
  SS       Mark  Pelczarski      SS
  SS     and  Paul Marentette    SS
  SS      Copyright (c) 1982     SS
  SS SoftSide Publications, Inc SS
  SS                             SS
  SS SS SS SS SS SS SS SS SS SS SS

110 NO=0:N1=1:N2=N1+N1:N3=N1+N2:N4=N2+
N2:N6=N3+N3:N7=N1+N6:N8=N6+N2:N14=N8+N
6:N17=N14+N3:N125=125:CL=13:RA=NO
120 DIM A$(260):DIM FI$(N14),FN$(N8),F
H$(N14),FMT$(N14):CH=NO:DL=156
130 DIM C1(N7),C2(N7),F$(400),B$(100),
C$(100):NI=-N1:FI$="D:":FH$="D:"
131 DIM SRT$(215):FOR I=N1 TO 215:READ
 A:SRT$(I)=CHR$(A):NEXT I
132 DATA 104,201,5,240,15,168,240,5,10
4,104,136,208,251,132,213,169,1,133,21
2,96,104,133,216,104,133,215,104
133 DATA 141,251,6,104,141,250,6,104,1
04,133,218,104,133,213,104,133,212,104
,104,133,217,169,0,133,209,133,214
134 DATA 162,1,165,215,133,205,165,216
,133,206,173,250,6,133,222,173,251,6,1
33,223,24,165,205,133,203,101,218
135 DATA 133,205,165,206,133,204,105,0
,133,206,24,165,222,133,220,105,3,133,
222,165,223,133,221,105,0,133,223
136 DATA 164,207,169,1,208,2,208,188,1
97,217,240,10,177,205,209,203,144,21,2
40,12,176,55,177,205,209,203,144
137 DATA 49,240,2,176,7,196,208,176,41
,200,144,219,169,1,133,209,164,218,136
,177,205,72,177,203,145,205,104
138 DATA 145,203,192,0,208,241,160,3,1
36,177,222,72,177,220,145,222,104,145,
220,192,0,208,241,232,224,0,208
139 DATA 2,230,214,228,212,208,134,165
,213,197,214,208,128,165,209,201,0,208
,162,169,0,133,212,133,213,96
140 OPEN #N2,N4,NO,"K:":GRAPHICS NO:SE
TCOLOR N2,CL,N4:SETCOLOR N4,CL,N4:GOTO
 290
150 FOR W=N1 TO 400:NEXT W:RETURN
160 REM GET RANDOM ACCESS POINTER
```

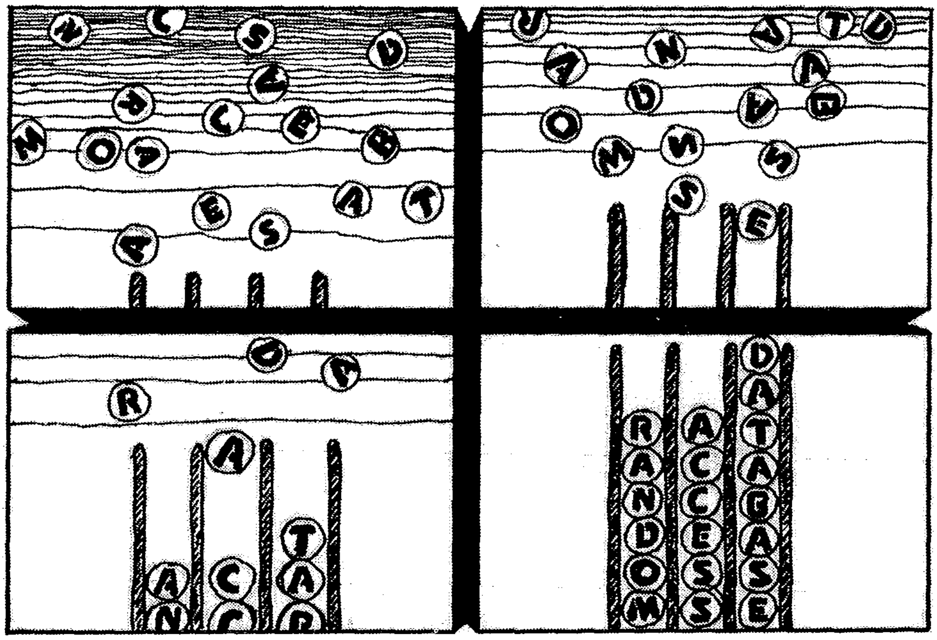
```
170 SEC=ASC(P$(R*N3+N1,R*N3+N1))*256+A
SC(P$(R*N3+N2,R*N3+N2)):BYT=ASC(P$(R*N
3+N3,R*N3+N3))
180 POINT #N1,SEC,BYT:RETURN
190 TI$=" ":TI$(B(NH+N1)-N2)=" ":TI$(N
2)=TI$:RETURN :REM BLANK RECORD
200 POSITION N6,22:FN$="":? CHR$(DL);"
ENTER FILE NAME ";:INPUT FN$:IF FN$=""
 THEN 240
210 IF FN$(N1,N1)<"A" OR FN$(N1,N1)>"Z
" THEN 240
220 FI$(N3)=FN$:FI$(LEN(FI$)+N1)=".DAT
":FOR I=N1 TO LEN(FN$):IF FN$(I,I)="."
 THEN 240
230 NEXT I:FH$(N3)=FN$:FH$(LEN(FH$)+N1
)=".HDG":RETURN
240 POSITION N6,22:? CHR$(DL);"INVALID
 NAME";:GOSUB 150:GOTO 200
249 REM Lines 250-260: Replace lower
case with the corresponding control
character.
250 ? CHR$(125);"brrrrrrrrrrrrrrrrv
 FILE: ";FN$
260 ? "b** DATABASE II **v   RECS: ";N
I+N1:? "brrrrrrrrrrrrrrrrrv";
270 IF RA>NO THEN ? "   FREE: ";MX-NI-
N1;
280 ? :? :? :RETURN
290 FN$="NONE":GOSUB 250:? "(I) INITIA
LIZE A NEW FILE"
300 ? "(L) LOAD A SAVED FILE?":? :? "S
ELECT? ";:GET #N2,A:IF CHR$(A)="I" THE
N GOSUB 860:GOTO 340
310 IF CHR$(A)<>"L" THEN 290
320 GOSUB 600:IF JJ=NO THEN 290
330 GOSUB 200:GOSUB 720
340 IF JJ>NO THEN 360
350 ? :? "FOR WARM RESTART: GOTO 360":
STOP
360 TRAP 360:CLOSE #N3:GOSUB 250:? "(S
) SAVE CURRENT FILE"
370 ? "(P) PRINT DATA"
380 ? "(A) ADD DATA"
390 ? "(C) CHANGE A RECORD"
400 ? "(D) DELETE A RECORD"
410 ? "(T) SORT"
420 ? "(F) LIST FILE NAMES"
430 ? "(N) START NEW FILE"
440 ? "(Q) QUIT":?
450 ? :? :? "SELECT ";
460 GET #N2,A:? CHR$(A)
470 IF CHR$(A)="S" THEN GOSUB 1180:GOT
O 360
480 IF CHR$(A)="P" THEN GOSUB 1280:GOT
O 360
490 IF CHR$(A)="A" THEN GOSUB 1600:GOT
O 360
500 IF CHR$(A)="C" THEN SB=N3:GOSUB 21
80:GOTO 360
510 IF CHR$(A)="D" THEN SB=N4:FS=N1:GO
SUB 2180:GOTO 360
520 IF CHR$(A)="T" THEN GOSUB 1970:GOT
O 360
530 IF CHR$(A)="F" THEN GOSUB 600:GOTO
 360
540 IF CHR$(A)="Q" OR CHR$(A)="N" THEN
 560
550 GOTO 360
560 IF SS THEN 580
570 GOSUB 1180
580 CLOSE #N1:IF CHR$(A)="N" THEN CLR
:RUN
590 TRAP 40000:GRAPHICS NO:END
600 POSITION N2,16:? CHR$(DL);"DATA FI
LES:":TRAP 610:OPEN #N3,N6,NO,"D:*.DAT
":GOTO 620:JJ=NO
610 POSITION N4,20:? "CAN'T READ DISK
DIRECTORY.":CLOSE #N3:GOSUB 150:RETURN
620 INPUT #N3,A$
630 FOR I=N17 TO 20:FOR J=N2 TO 29 STE
P 9:IF LEN(A$)<N17 THEN 680
640 POSITION J,I:? A$(N3,10):JJ=JJ+N1:
INPUT #N3,A$:NEXT J:NEXT I
650 IF LEN(A$)<N17 THEN 680
660 POSITION 8,23:? "PRESS RETURN FOR
MORE ";:GET #N2,A
670 FOR I=23 TO N17 STEP -N1:POSITION
NO,I:? CHR$(DL);:NEXT I:GOTO 630
680 TRAP 360:IF JJ=NO THEN ? :? "NO DA
TA FILES IN DIRECTORY";:GOSUB 150
690 CLOSE #N3:POSITION N6,23:IF CHR$(A
)<>"L" THEN ? "PRESS RETURN TO CONTINU
E ";:GET #N2,A
700 RETURN
710 REM LOAD SUBROUTINE
720 JJ=N1:GOSUB 250:? :? "LOADING ";FN
$;:TRAP 840:OPEN #N3,N4,N0,FH$
730 INPUT #N3,NH,NI,HL,IL,MX,BLK,RA
740 DIM H$(HL*(NH+1)),B(NH+1),I$(MX*IL
+N1),P$(MX*N3),T$(NH*IL+IL)
750 FOR I=N0 TO NH+N1:INPUT #N3,J:B(I)
=J:NEXT I:DIM TI$(B(NH+1)-N1):I$=" ":I
$(MX*IL-N1)=" ":I$(N2)=I$:SEG=N1
760 INPUT #N3,A$:IF LEN(A$)=N0 THEN 78
0
770 H$(SEG)=A$:SEG=SEG+250:GOTO 760
780 INPUT #N3,PL:FOR I=N1 TO PL:GET #N
3,BYT:P$(I)=CHR$(BYT):NEXT I
790 CLOSE #N3:OPEN #N1,12,N0,FI$:GOSUB
 820
800 SS=N1:RETURN
810 REM READ ITEMS UNDER HEADING CH
820 S=B(CH):E=B(CH+N1)-N1:L=E-S+N1:FOR
 I=N0 TO NI:R=I:GOSUB 170:INPUT #N1,TI
$
830 I$(I*L+N1,I*L+L)=TI$(S,E):NEXT I:R
ETURN
840 ? CHR$(156);CHR$(253);"DISK ACCESS
 ERROR";:CLOSE #N1:CLOSE #N3:JJ=N0:GOS
UB 150:POP :GOTO 340
850 REM INITIALIZE SUBROUTINE V.2
860 GOSUB 200
870 GOSUB 250:? :? "HOW MANY HEADINGS
";:TRAP 870:INPUT NH:IF NH<N1 THEN 870
880 ? :? "ENTER MAXIMUM HEADING LENGTH
 ";:TRAP 880:INPUT HL:IF HL<N1 THEN 88
0
890 TRAP 1120:DIM H$(HL*NH),B(NH):NH=N
H-N1:NI=-N1:H$="":B(0)=N1:IL=N0:GOSUB
250
900 FOR I=N0 TO NH
910 ? :? "HEADING #";I+N1;" :";:INPUT
A$:J=LEN(A$):IF J>HL THEN ? "MAXIMUM L
ENGTH IS ";HL;". REENTER":GOTO 910
915 IF J=N0 THEN 910
920 IF LEN(A$)<HL THEN A$(LEN(A$)+N1)=
" ":GOTO 920
930 TRAP 930:? "MAXIMUM LENGTH OF ";A$
(N1,J);": ";:INPUT J:TRAP 1120:B(I+1)=
B(I)+J:IF J>IL THEN IL=J
940 H$(LEN(H$)+N1)=A$:NEXT I
950 MX=INT((FRE(0)-2000)/(IL+N3)):TRAP
 840:OPEN #N3,6,N0,"D:*.*":REM FIND SP
ACE ON DISK
960 INPUT #N3,A$:IF LEN(A$)<N17 THEN 9
80
970 GOTO 960
980 CLOSE #N3:TRAP 1120:FREE=VAL(A$(N1
,3))*N125-((NH+1)*HL):FREE=INT(FREE/(B
(NH+1)+N3)):IF FREE<MX THEN MX=FREE
990 GOSUB 250:? "DISK SPACE REPORT:"
1000 ? :? "DATABASE CAPACITY = ";MX;"
RECORDS":LRECL=B(NH+N1):BLK=50:LO=N125
:HI=BLK:IF HI>MX THEN HI=MX
1010 FOR I=HI TO N1 STEP -N1:IF LRECL*
I<SEC THEN 1030
1020 FREE=LRECL*I/N125-INT(LRECL*I/N12
5):IF FREE<LO THEN LO=FREE:BLK=I
1030 NEXT I:? :? "BLOCK SIZE = ";BLK;"
 RECORDS."
1040 ? :? "SECTORS/BLK = ";INT(BLK*LRE
CL/N125+0.5):GOSUB 150
1050 DIM I$(MX*IL+N1),P$(MX*N3),TI$(B(
NH+1)-N1),T$(NH*IL+IL):I$=" ":I$(MX*IL
-N1)=" ":I$(N2)=I$:GOSUB 150
1060 IO=8:R=N0:E=BLK-N1:GOSUB 1070:JJ=
N1:RETURN
1070 GOSUB 190:TRAP 840:? :? :? "ONE M
OMENT...ALLOCATING DISK SPACE":? "FOR
RECORDS ";R+1;" TO ";E+1;"..."
1080 CLOSE #N1:OPEN #N1,IO,N0,FI$:FOR
I=R TO E:NOTE #N1,SEC,BYT:HI=INT(SEC/2
56):LO=SEC-HI*256:S=I*N3
1090 P$(S+N1,S+N1)=CHR$(HI):P$(S+N2,S+
N2)=CHR$(LO):P$(S+N3,S+N3)=CHR$(BYT)
1100 PRINT #N1;TI$:NEXT I:CLOSE #N1:OP
EN #N1,12,N0,FI$:RA=E
1110 SS=N0:RETURN
1120 ? :? "ERROR - FILE NOT INITIALIZE
D":JJ=N0:GOSUB 150:RETURN
1130 REM WRITE RECORD R FROM TI$
1140 TRAP 840:GOSUB 170:PRINT #N1;TI$:
TRAP 360:RETURN
```

```
1150 REM READ RECORD R INTO TI$
1160 TRAP 840:R=I:GOSUB 170:INPUT #N1,
TI$:TRAP 360:RETURN
1170 REM WRITE HDG FILE
1180 JJ=N1:GOSUB 250:? :? "SAVING ";FN
$;:TRAP 840:OPEN #N3,N8,N0,FH$
1190 PRINT #N3;NH:PRINT #N3;NI:PRINT #
N3;HL:PRINT #N3;IL:PRINT #N3;MX:PRINT
#N3;BLK:PRINT #N3;RA
1200 FOR I=N0 TO NH+N1:PRINT #N3;B(I):
NEXT I:SEG=N1
1210 IF LEN(H$)<=SEG+249 THEN PRINT #N
3;H$(SEG,LEN(H$)):GOTO 1240
1220 PRINT #N3;H$(SEG,SEG+249):SEG=SEG
+250:IF SEG>LEN(H$) THEN 1240
1230 GOTO 1210
1240 PRINT #N3;"":PL=LEN(P$):PRINT #N3
;PL
1250 FOR I=N1 TO PL:BYT=ASC(P$(I)):PUT
 #N3,BYT:NEXT I
1260 CLOSE #N3:SS=N1:RETURN
1270 REM PRINT SUBROUTINE VERS.3
1280 FS=N1:IF NI=-N1 THEN GOSUB 2850:R
ETURN
1290 GOSUB 250:? "FORMATTED OUTPUT? (Y
/N) ";:GET #N2,A:? :IF CHR$(A)="Y" THE
N GOSUB 2870:TRAP 360:FS=N2:GOTO 1320
1300 IF CHR$(A)<>"N" THEN 1280
1310 FS=N1:GOTO 1330
1320 IF FS=N2 AND F$="" THEN 1280
1330 GOSUB 250:SB=N1:? "(S) SCREEN or"
:? "(P) PRINTER ?";:GET #2,A:? :IF CHR
$(A)="P" THEN SB=N2:GOTO 1350
1340 IF CHR$(A)<>"S" THEN 1330
1350 GOSUB 2190:CLOSE #N3:POKE 752,N0:
RETURN
1360 REM PRINT ONE RECORD VERS. 4
1370 ? CHR$(125);:ON FS GOSUB 1430,146
0:POSITION N2,22:? "PRESS RETURN (ESC
FOR MENU)";
1380 GET #N2,A:IF A=27 THEN RS=N1
1390 RETURN
1400 REM PRINT ONE RECORD TO PRINTER,
VERS.4
1410 ON FS GOSUB 1420,1460:RETURN
1420 IF SB=N4 THEN ? CHR$(125);
1430 PRINT #N3:PRINT #N3;"RECORD ";I+N
1:PRINT #N3:FOR J=N0 TO NH:S=B(J):E=B(
J+1)-N1
1440 PRINT #N3;H$(J*HL+N1,J*HL+HL);" "
;TI$(S,E):NEXT J:RETURN
1450 REM PRINT ONE FORMAT
1460 J=N1:T=N0:B$="":IF F$(LEN(F$))<>"
6" THEN F$(LEN(F$))="6"
1470 J1=VAL(F$(J,J)):J=J+N1:IF J1<5 TH
EN N=VAL(F$(J,J+N1)):J=J+N2
1480 ON J1 GOTO 1490,1500,1510,1520,15
40,1590
1490 A$=H$(N*HL+N1,N*HL+HL):GOTO 1570
1500 S=B(N):E=B(N+N1)-N1:L=E-S+N1:A$=T
I$(S,E):GOTO 1570
1510 FOR QQ=N1 TO N:B$(LEN(B$)+N1)=" "
:NEXT QQ:GOTO 1580
1520 PRINT #N3;B$:IF N>N1 THEN FOR J2=
N2 TO N:PRINT #N3:NEXT J2
1530 B$="":GOTO 1580
1540 J2=J
1550 IF F$(J2,J2)<>"!" THEN J2=J2+N1:G
OTO 1550
1560 A$=F$(J,J+(J2-J)-N1):J=J2+N1
1570 B$(LEN(B$)+N1)=A$
1580 GOTO 1470
1590 PRINT #N3;B$:RETURN
1600 SS=N0:IF NI+N1=MX THEN ? :? "FILE
 IS FULL":GOSUB 150:RETURN
1610 R=NI+N1:IF R<=RA THEN 1640
1620 GOSUB 250:E=RA+BLK:IF E>MX-N1 THE
N E=MX-N1
1630 IO=9:GOSUB 1070
1640 GOSUB 250:? :? "RECORD NUMBER ";N
I+N2;"  (ENTER ! TO END)":? :? :FOR J=
N0 TO NH:GOSUB 1710
1650 IF A$(N1,N1)="!" THEN J=NH:NEXT J
:RETURN
1660 NEXT J:NI=NI+N1
1670 GOSUB 1140:S=B(CH):E=B(CH+N1)-N1:
L=E-S+N1
1680 I$(NI*L+N1,NI*L+L)=TI$(S,E)
1690 GOTO 1600
1700 REM DATA INPUT SUBROUTINE VERS.1
```

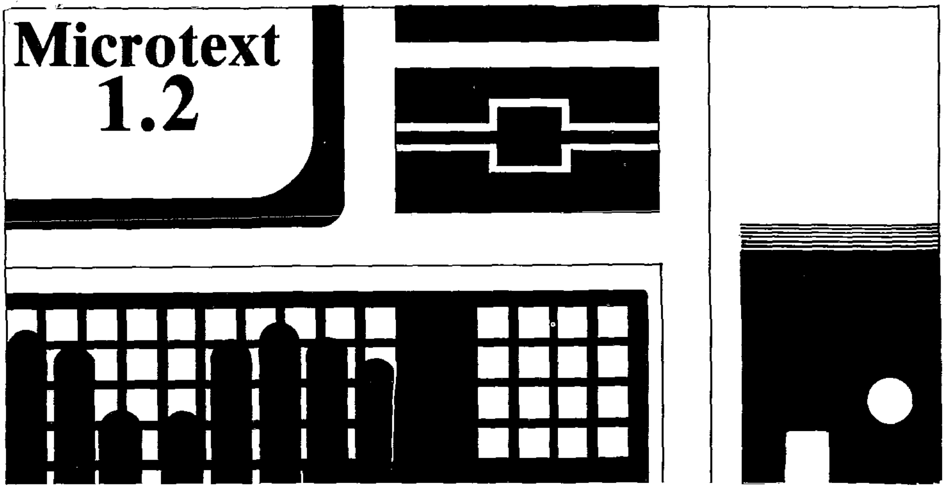
```
1710 ? H$(J*HL+1,J*HL+HL);" : ";:INPUT
 A$:S=B(J):E=B(J+1)-N1:L=E-S+N1
1720 IF LEN(A$)<L THEN A$(LEN(A$)+N1)=
" ":GOTO 1720
1730 TI$(S,E)=A$:RETURN
1740 REM CHANGE SUBROUTINE VERS.2
1750 GOSUB 250:? "(C) CHANGE ITEM, (K)
 KEEP ITEM, OR":? "(R) KEEP REMAINDER
OF RECORD"
1760 ? :? "RECORD ";I+N1:CS=N1:RS=N0:F
OR J=0 TO NH:? :S=B(J):E=B(J+N1)-N1:L=
E-S+N1
1770 ? H$(J*HL+N1,J*HL+HL);" : ";TI$(S
,E);" ?";
1780 IF RS=N1 THEN PRINT :GOTO 1840
1790 GET #N2,A:IF CHR$(A)<>"C" AND CHR
$(A)<>"K" AND CHR$(A)<>"R" THEN 1790
1800 ? CHR$(A):IF CHR$(A)="K" THEN 184
0
1810 IF CHR$(A)="R" THEN RS=N1:GOTO 18
40
1820 POKE 752,N0:GOSUB 1710:POKE 752,N
1
1830 CS=N0
1840 NEXT J:RS=N0:IF CS=N0 THEN R=I:GO
SUB 1140:S=B(CH):E=B(CH+N1)-N1:L=E-S+N
1:I$(I*L+N1,I*L+L)=TI$(S,E)
1850 RETURN
1860 REM DELETE SUBROUTINE VERS.2
1870 ? :POKE 752,N0:? "DELETE THIS REC
ORD? ";
1880 GET #N2,A:IF CHR$(A)<>"Y" AND CHR
$(A)<>"N" THEN 1880
1890 ? CHR$(A):POKE 752,N1:IF CHR$(A)=
"N" THEN 1950
1900 A$=P$(I*N3+N1,I*N3+N3):P$(I*N3+N1
)=P$(I*N3+N1+N3):P$(RA*N3+N1)=A$
1910 I$(LEN(I$)+N1)=" "
1920 L=B(CH+N1)-B(CH):I$(I*L+N1)=I$(I*
L+L+N1):IF LEN(I$)=N1 THEN I$="":GOTO
1940
1930 I$=I$(N1,LEN(I$)-N1)
1940 NI=NI-N1:SS=N0:I=I-N1
1950 RETURN
1960 REM SORT ROUTINE
1970 IF NI=-N1 THEN GOSUB 2850:RETURN
1980 GOSUB 250:FOR J=N0 TO NH:? "(";J+
N1;") ";H$(J*HL+N1,J*HL+HL):NEXT J
1990 ? :? "SORT WHICH HEADING ";:INPUT
 J1:J1=J1-N1
2000 IF J1<N0 OR J1>NH THEN RETURN
2010 IF J1<>CH THEN CH=J1:GOSUB 820
2020 GOSUB 250:? "(A) ASCENDING, OR (D
) DESCENDING":GET #N2,A:IF CHR$(A)="A"
 THEN A=N0:GOTO 2050
2030 IF CHR$(A)="D" THEN A=N1:GOTO 205
0
2040 GOTO 2020
2050 ? :? "SORTING...";:POKE 207,N0:PO
KE 208,L-N1:T=USR(ADR(SRT$),ADR(I$),AD
R(P$),L,NI+N1,A)
2160 ? "COMPLETED.":GOSUB 150:SS=N0:RE
TURN
2170 REM SEARCH SUBROUTINE, VERS. 2
2180 IF NI=-N1 THEN GOSUB 2850:RETURN
2190 I1=N0:I2=NI:J=N0:C1(N0)=-N1:BS=N1
:NF=N1
2200 GOSUB 250:? "SEARCH CRITERIA:":?
:? "0) RECORD NUMBER"
2210 FOR I=N0 TO NH:PRINT I+N1;") ";H$
(I*HL+N1,I*HL+HL):NEXT I:? :? NH+N2;")
 BEGIN"
2220 POSITION N2,20:? "WHICH FIELD: ";
:INPUT I:IF I<N0 OR I>NH+N2 THEN 2220
2230 IF I=NH+N2 THEN C1(J)=-N1:GOTO 23
70
2240 C1(J)=I-N1
2250 POSITION N2,21:? "(1) <=      (2)
 =      (3) >=    ";:INPUT A:C2(J)=A:I
F C2(J)<N1 OR C2(J)>N3 THEN 2250
2260 POSITION N2,22:? "VALUE:";:IF C1(
J)=-N1 THEN 2310
2270 PRINT " ";:INPUT A$
2280 IF LEN(A$)<IL THEN A$(LEN(A$)+N1)
=" ":GOTO 2280
2290 C$(J*IL+N1,J*IL+IL)=A$:J=J+N1:IF
J>N7 THEN 2380
2300 GOTO 2200
2310 PRINT " ";:INPUT I:IF I<N1 OR I>N
I+N1 THEN 2310
```

```
2320 I=I-N1
2330 IF C2(J)=N1 THEN I2=I
2340 IF C2(J)=N2 THEN I1=I:I2=I
2350 IF C2(J)=N3 THEN I1=I
2360 GOTO 2200
2370 IF J<N2 THEN 2400
2380 POSITION N2,21:? CHR$(DL);"1) ITE
M MUST MEET ALL CONDITIONS":? CHR$(DL)
;"2) ITEM MAY MEET ANY CONDITION ";:IN
PUT BS
2390 IF BS<N1 OR BS>N2 THEN 2380
2400 RS=NO:J1=C1(NO):DS=NO:FOR J=NO TO
 7:IF C1(J)=-N1 THEN J=7:GOTO 2420
2410 IF J1<>C1(J) THEN J1=-N2
2420 NEXT J
2430 IF J1>-N1 AND J1<>CH THEN CH=J1:G
OSUB 820
2440 IF J1=-N2 THEN DS=N1
2450 TRAP 2480:IF SB=N2 THEN OPEN #N3,
N8,NO,"P:":TRAP 360:GOTO 2490
2460 OPEN #N3,N8,NO,"E:":POKE 752,N1:T
RAP 360:SETCOLOR N2,CL,N4:SETCOLOR N4,
CL,N4
2470 POSITION N2,22:? "SEARCHING...";:
GOTO 2490
2480 CLOSE #N3:? "PRINTER NOT ON LINE"
:GOTO 2840
2490 I=I1-N1:FOR I9=I1 TO I2:I=I+N1
2500 IF DS=NO THEN S=B(CH):E=B(CH+N1)-
N1:L=E-S+N1:TI$(S,E)=I$(I*L+N1,I*L+L):
GOTO 2520
2510 GOSUB 1160
2520 AS=NO:FOR J=NO TO N7
2530 IF C1(J)=-N1 THEN J=N7:GOTO 2720
2540 S=B(C1(J)):E=B(C1(J)+N1)-N1:L=E-S
+N1:B$=C$(J*IL+N1,J*IL+L):A$=TI$(S,E)
2550 FOR T=L TO N1 STEP -N1:IF A$(T,T)
=" " THEN NEXT T:A$="":GOTO 2570
2560 A$=A$(N1,T)
2570 FOR T=L TO N1 STEP -N1:IF B$(T,T)
=" " THEN NEXT T:B$="":GOTO 2590
2580 B$=B$(N1,T)
2590 ON C2(J) GOTO 2600,2620,2670
2600 IF A$<=B$ THEN 2690
2610 GOTO 2710
2620 IF A$=B$ THEN 2690
2630 IF T=NO THEN 2710
2640 IF B$(T)<>"*" THEN 2710
2650 IF A$(N1,T-N1)=B$(N1,T-N1) THEN 2
690
2660 GOTO 2710
2670 IF A$>=B$ THEN 2690
2680 GOTO 2710
2690 IF BS=N2 THEN AS=N1:J=N7
2700 GOTO 2720
2710 IF BS=N1 THEN AS=N2:J=N7
2720 NEXT J
2730 IF AS=NO AND BS=N1 THEN 2750
2740 IF AS=NO OR AS=N2 THEN 2810
2750 IF DS=NO THEN GOSUB 1160
2760 NF=0:IF SB=1 THEN GOSUB 1370
2770 IF SB=N2 OR SB=N4 THEN GOSUB 1410
2780 IF SB=N3 THEN GOSUB 1750
2790 IF SB=N4 THEN GOSUB 1870
2800 IF RS=N1 THEN I9=I2
2810 POSITION N2,22:? "SEARCHING...
                    ";:NEXT I9:POKE 7
52,NO:IF RS THEN RETURN
2820 IF NF THEN POSITION N2,22:? "NO M
ATCHES FOUND...";:GOTO 2840
2830 POSITION N2,22:? "END OF FILE..."
;
2840 ? CHR$(253);"RETURN FOR MENU";:GE
T #N2,A:RETURN
2850 ? CHR$(253);"NO DATA IN MEMORY.":
GOSUB 150:RETURN
2860 REM PRINT FORMATTING, V.1
2870 GOSUB 250:IF F$="" THEN 2900
2880 ? "SAME FORMAT? ";:GET #N2,A:PRIN
T :IF CHR$(A)="Y" THEN RETURN
2890 IF CHR$(A)<>"N" THEN 2870
2900 ? "(L) LOAD FORMAT or":? "(C) CRE
ATE FORMAT ";:GET #N2,A:? :IF CHR$(A)=
"C" THEN 2970
2910 IF CHR$(A)<>"L" THEN 2870
2920 F$="":GOSUB 3180:IF FMT$="D:" THE
N RETURN
2930 TRAP 2950:OPEN #N3,N4,NO,FMT$
2940 INPUT #N3,NF:FOR J=NO TO NF:INPUT
 #N3,A$:F$(LEN(F$)+N1)=A$:NEXT J:GOTO
```

```
2960
2950 ? "FORMAT ";FMT$;" NOT FOUND":GOS
UB 150:FMT$="":F$=""
2960 CLOSE #N3:RETURN
2970 TRAP 2970:NF=N0:J=N0:F$="":GOSUB
250:? "START IN THE UPPER LEFT CORNER
AND":? "WORK ACROSS EACH LINE.":?
2980 ? "1:HEADING":? "2:ITEM":? "3:TAB
":? "4:NEXT LINE":? "5:STRING":? "6:EN
D":?
2990 INPUT J1:IF J1<N1 OR J1>N6 THEN 2
990
3000 F$(LEN(F$)+N1)=STR$(J1):J=J+N1
3010 ON J1 GOTO 3020,3020,3050,3050,30
90,3110
3020 GOSUB 250:FOR T=N0 TO NH:PRINT T+
N1;") ";H$(T*HL+N1,T*HL+HL):NEXT T
3030 ? :? "WHICH? ";:INPUT T:T=T-N1:IF
 T<N0 OR T>NH THEN 3030
3040 GOTO 3060
3050 ? "HOW MANY? ";:INPUT T:IF T<N1 O
R T>99 THEN ? "OUT OF RANGE.":GOTO 305
0
3060 A$=STR$(T):IF T<10 THEN A$="0":A$
(LEN(A$)+N1)=STR$(T)
3070 F$(LEN(F$)+N1)=A$:J=J+N2
3080 GOTO 3100
3090 ? "STRING: ";:INPUT A$:A$(LEN(A$)
+N1)="!":F$(LEN(F$)+N1)=A$:J=J+LEN(A$)
3100 GOSUB 250:GOTO 2980
3110 TRAP 3170:GOSUB 3180:IF FMT$="D:"
 THEN F$="":RETURN
3120 OPEN #N3,N8,N0,FMT$
3130 PRINT #N3;INT((LEN(F$)-N1)/250)
3140 FOR J=N0 TO INT((LEN(F$)-N1)/250)
:IF LEN(F$)<J*250+250 THEN ? #N3;F$(J*
250+N1,LEN(F$)):GOTO 3160
3150 PRINT #N3;F$(J*250+N1,J*250+250):
NEXT J
3160 CLOSE #N3:RETURN
3170 ? "FORMAT OUTPUT ERROR":GOSUB 150
:CLOSE #N3:RETURN
3180 GOSUB 250:FMT$="D:":? "ENTER FORM
AT NAME ";:INPUT A$:J=LEN(A$):IF J<N1
THEN RETURN
3190 FOR I=N1 TO J:IF A$(I,I)="." THEN
 ? :? "INVALID NAME":GOSUB 150:GOTO 31
80
3200 NEXT I:FMT$(N3)=A$:FMT$(J+N3)=".F
MT":RETURN
```



Microtext 1.2



by Jon Voskuil

***Microtext 1.2* is a word-processing program for an Atari with 16K RAM. A printer is desirable.**

Upon running, *Microtext 1.2* displays a mostly blank screen with an instruction summary line at the top.You can either start typing, or load a previously saved file from disk or tape.

Holding down CTRL, and then pressing S, L, R, P, or E, will access the save, load, review, printout, or edit functions. Although not mentioned in the command summary on the screen, pressing CTRL-Q will quit the program.

Saving and loading files is simply a matter of answering the questions about the medium to be used (tape or disk) and, if disk, the file name. Once you have entered a file name, it will be used as the default until you specify another one or exit the program: Just press RETURN when asked for the file name. This simplifies repeated saves during entry of a long document.

The review function causes the computer to return to the beginning of the text in memory and scroll through it to the end. During this scrolling, you can press any key to pause. Then, pressing the spacebar will cause one or more lines to be displayed; pressing RETURN will cause the scrolling to continue; and pressing E will enter the editing mode.

In the editing mode, you can move a cursor up and down through your text, to locate any line which you want to edit or delete. This movement is accomplished with the up- and down-arrow keys. You have four options while in the editing mode: Pressing ESC will exit to the review mode; pressing D will delete the line at the cursor; pressing X will delete everything from the cursor to the end of the text; and pressing RETURN will allow you to edit the line at the cursor.

If you choose to edit a line, the screen will first clear, and then display a number of lines of text with a gap of several lines in the middle. The cursor will be positioned at the beginning of the line you have chosen to edit, and you can proceed to type in a new line to replace the old one. The new one can be shorter than the original, or may occupy multiple screen lines. Any part of the original line that you want retained must be retyped: Whatever you type in will replace the entire line. When you have finished entering the new text, press CTRL-F (not RETURN, unless you want a carriage return in the text itself). The computer will determine whether the text lines need to be rearranged, and then return you to the review mode.

The printout function allows you to send your text to a printer, after selecting margins, and line spacing. Pressing RETURN in response to the offered options will select the default value indicated.

Program Notes

The DIMension statement in line 120 reserves memory for the strings which hold the text. The numbers listed work for 40K system with DOS booted, but will need to be adjusted downward if your system has less available memory. All REM lines may be deleted without affecting the program's operation, to gain more memory space. Many of the individual program lines could also be squeezed together on multiple-statement lines; this kind of packing has been avoided for the sake of clarity, but would increase available memory and possibly execution speed. Of course, you should first verify your typing with *SWAT* before you make any changes.

```
 SS SS SS SS SS SS SS SS SS SS SS
 SS                            SS
 SS        Atari  BASIC        SS
 SS      'Microtext  1.2'      SS
 SS   Author: Jon R. Voskuil   SS
 SS     Copyright (c) 1982     SS
 SS SoftSide Publications, Inc SS
 SS                            SS
 SS SS SS SS SS SS SS SS SS SS SS
```

```
10 DIM CL$(1):CL$=CHR$(125):PRINT CL$
15 POKE 752,1
20 POSITION 8,8:PRINT "M I C R O T E X
 T   1 . 2"
30 POSITION 12,12:PRINT "BY JON R. VOS
KUIL"
40 PRINT :PRINT "COPYRIGHT 1982  SOFTS
IDE PUBLICATIONS"
50 FOR Z=1 TO 1000:NEXT Z
```

Initialization.

```
99 REM INITIALIZATION
100 PRINT CL$:TRAP 20000
120 DIM L$(40),T$(14000),B$(1),C$(1),C
R$(1),LIN$(37),LNXT$(40),F$(14),S$(37)
,FT$(14),LP(500),TT$(40),X$(5)
125 DIM P$(255),PP$(80),Q$(37),F1$(14)
130 BKSP=126:RTN=155:SPC=32:B$=CHR$(12
6):CR$=CHR$(20)
135 T$="":L$="":LP(0)=0
140 CHAR=1
150 LN=1
160 LIN$="----------------------------
---------":S$="

     "
170 OPEN #1,4,0,"K:"
180 LWID=36
185 POKE 752,0
190 V=2:H=2
200 POSITION 2,0:PRINT "^SAVE  ^LOAD
^REVIEW  ^EDIT  ^PRINT";
210 POSITION 2,1:PRINT LIN$;
220 POSITION H,V:PRINT " ";B$;
```

Input loop.

```
499 REM INPUT LOOP
500 GET #1,C:C$=CHR$(C)
520 IF C=RTN THEN C$=CR$
639 REM BACKSPACE
640 IF C<>BKSP THEN 720
650 IF CHAR<2 THEN L$="":GOTO 500
660 PRINT B$;" ";B$;
670 CHAR=CHAR-1
675 IF CHAR<2 THEN 500
680 L$=L$(1,CHAR-1)
700 GOTO 500
719 REM CHK FOR CTRL CHAR
720 IF C<SPC THEN 2000
739 REM CHK FOR END OF LN
740 CHAR=CHAR+1:IF CHAR<LWID OR C=SPC
OR C=RTN THEN 760
```

```
750 GOSUB 1000
759 REM ADD TO STRING
760 L$(CHAR-1)=C$
779 REM RETURN
780 IF C<>RTN THEN 880
820 PRINT C$
840 GOSUB 6000:SLOC=0
850 CHAR=1
860 GOTO 500
879 REM PRINT ON SCRN
880 PRINT C$;
899 REM UPDATE SPC PNTER, CHK FOR END
OF LINE
900 IF C=32 THEN SLOC=CHAR-1:IF CHAR=L
WID THEN GOSUB 6000:CHAR=1:SLOC=0:PRIN
T
920 GOTO 500
```

Subroutine to break line at a space and to initialize the next line.

```
1000 IF SLOC=0 THEN PRINT :GOTO 1100
1020 SS=LWID-SLOC-1:FOR J=1 TO SS:PRIN
T B$;:NEXT J
1040 FOR J=1 TO SS:PRINT " ";:NEXT J:P
RINT
1050 IF SLOC=LEN(L$) THEN LNXT$="":GOT
O 1100
1060 LNXT$=L$(SLOC+1)
1080 L$=L$(1,SLOC)
1100 GOSUB 6000
1110 L$=LNXT$
1115 LNXT$=""
1120 PRINT L$;
1140 CHAR=LEN(L$)+2
1150 SLOC=0
1160 RETURN
```

Subroutine to process command codes.

```
1999 REM PROCESS CTRL CHAR
2000 V=PEEK(84):H=PEEK(85)
2050 TL=LEN(T$)
2060 T$(TL+1)=L$
2070 POSITION 2,0:PRINT S$:PRINT LIN$;
2080 IF C<>6 THEN 2100:REM CTRL-F (BAC
K TO EDIT)
2085 IF CHAR>1 THEN LP(LN)=LEN(T$):LN=
LN+1:LP(LN)=LEN(T$):L$=""
2090 RETURN
2100 IF C=18 THEN GOSUB 3000:REM CTL-R
2200 IF C=19 THEN GOSUB 4000:REM CTL-S
2300 IF C=12 THEN GOSUB 5000:REM CTL-L
2400 IF C=17 THEN END :REM CTL-Q
2500 IF C=16 THEN GOSUB 7000:REM CTL-P
2600 IF C=5 AND LN>1 THEN I=LN-1:VV=V:
POSITION 2,VV-1:GOSUB 9000:GOSUB 3000:
REM CTL-E
2900 IF TL>0 THEN T$=T$(1,TL)
2950 GOTO 200
```

Subroutine to review text. Line 3055, a brief delay loop, may optionally be deleted.

```
2999 REM REVIEW TEXT
3000 PRINT CL$;"Press any key to pause
":PRINT LIN$
3040 IF LN=1 THEN 3210
3050 FOR I=1 TO LN-1
3055 FOR Z=1 TO 20:NEXT Z
3060 PRINT T$(LP(I-1)+1,LP(I))
3070 IF PEEK(764)=255 AND  NOT STP THE
N 3200
3080 STP=0:POKE 764,255
3090 VV=PEEK(84)
3100 POSITION 2,1:PRINT LIN$;:POSITION
 2,0:PRINT "  RTN:Cont     SPC:Stp
 E:Edit   ";
3120 X=PEEK(764):POKE 764,255
3125 IF X=42 THEN GOSUB 9000:GOTO 3000
3160 IF X=12 THEN 3190
3170 IF X<>33 THEN 3120
3180 STP=1
3190 POSITION 2,VV:POKE 764,255
3200 NEXT I
3210 X=LP(LN-1)+1:IF X<=LEN(T$) THEN P
RINT T$(X);
3220 H=PEEK(85):V=PEEK(84)
3230 RETURN
```

Subroutine to save to tape or disk.

```
3999 REM SAVE TO DISK/TAPE
```

```
4000 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "Save to Tape or Disk? (T/D/E
SC) ";
4020 GET #1,X
4030 IF X=27 THEN 4400
4060 IF X=84 THEN 4200
4070 IF X<>68 THEN 4000
4075 F1$=F$
4080 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "File Name: ";:INPUT F$
4082 IF F$="" AND F1$="" THEN 4080
4083 IF F$="" THEN F$=F1$
4085 IF F$(1,1)<>"D" THEN FT$="D:":FT$
(3)=F$:F$=FT$
4090 POSITION 2,0:PRINT "Insert disk a
nd press RETURN";:GET #1,X
4100 OPEN #2,8,0,F$:GOTO 4210
4200 POSITION 2,0:PRINT "Start tape re
corder and press RETURN";:GET #1,X
4205 OPEN #2,8,0,"C:"
4210 PRINT #2;LN:PRINT #2;SLOC:PRINT #
2;CHAR
4220 FOR I=1 TO LN-1
4230 PRINT #2;T$(LP(I-1)+1,LP(I))
4240 NEXT I
4250 IF CHAR>1 THEN PRINT #2;T$(LP(LN-
1)+1)
4300 CLOSE #2
4400 RETURN
```

Subroutine to load text from tape or disk.

```
4999 REM LOAD FROM DISK/TAPE
5000 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "Load from Tape or Disk? (T/D
/ESC) ";
5020 GET #1,X
5030 IF X=27 THEN 5400
5060 IF X=84 THEN 5200
5070 IF X<>68 THEN 5000
5075 F1$=F$
5080 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "File Name: ";:INPUT F$
5082 IF F$="" AND F1$="" THEN 5080
5083 IF F$="" THEN F$=F1$
5085 IF F$(1,1)<>"D" THEN FT$="D:":FT$
(3)=F$:F$=FT$
5090 POSITION 2,0:PRINT "Insert disk a
nd press RETURN";:GET #1,X
5100 OPEN #2,4,0,F$:GOTO 5210
5200 POSITION 2,0:PRINT "Start tape re
corder and press RETURN";:GET #1,X
5205 OPEN #2,4,0,"C:"
5210 INPUT #2;LN:INPUT #2;SLOC:INPUT #
2;CHAR
5215 T$=""
5220 FOR I=1 TO LN-1
5230 INPUT #2,TT$:T$(LEN(T$)+1)=TT$
5240 LP(I)=LEN(T$)
5250 NEXT I
5255 TL=LEN(T$):L$=""
5260 IF CHAR>1 THEN INPUT #2,TT$:T$(LE
N(T$)+1)=TT$:LP(LN)=LEN(T$):L$=TT$
5300 CLOSE #2
5350 GOSUB 3000
5400 RETURN
```

Subroutine to add a line of text to the main text string.

```
5999 REM ADD LN TO TEXT STRING
6000 T$(LEN(T$)+1)=L$
6080 LP(LN)=LEN(T$)
6100 LN=LN+1:LP(LN)=LEN(T$)
6150 L$=""
6200 RETURN
```

Subroutine to print the text file in memory to a printer.

```
6999 REM PRINTOUT RTN
7000 PRINT CL$:POSITION 2,6:LIN=0
7010 ? "Left margin? (Default = 10) ";
:INPUT X$:LM=10:IF LEN(X$)>0 THEN IF V
AL(X$)>0 THEN LM=VAL(X$):IF LM>37 THEN
 LM=37
7020 PRINT :PRINT "Right margin? (Defa
ult = 70) ";:INPUT X$:RM=70:IF LEN(X$)
>0 THEN IF VAL(X$)>0 THEN RM=VAL(X$)
7030 PRINT :PRINT "Line spacing? (Defa
ult = 2) ";:INPUT X$:LS=2:IF LEN(X$)>0
 THEN IF VAL(X$)>0 THEN LS=VAL(X$)
```

```
7040 LL=RM-LM
7070 PRINT CL$:LPRINT "":P$="":CR=0:I=
0
7080 I=I+1:P$(LEN(P$)+1)=T$(LP(I-1)+1,
LP(I))
7090 IF P$(LEN(P$))=CR$ THEN CR=1:GOTO
 7110
7100 IF LEN(P$)<255-LWID AND I<LN-1 TH
EN 7080
7110 GOSUB 7500:CR=0
7120 IF I<LN-1 THEN 7080
7130 LPRINT S$(1,LM);L$;
7150 LPRINT ""
7160 GOSUB 3000
7170 RETURN
7500 L=LL
7510 IF LEN(P$)>LL THEN 7550
7520 IF ( NOT CR) THEN 7640
7530 LP=LEN(P$):IF LP<2 THEN PP$="":P$
="":GOTO 7590
7540 PP$=P$(1,LP-1):P$="":GOTO 7590
7550 C$=P$(L,L):IF C$=" " THEN 7580
7560 L=L-1:IF L>0 THEN 7550
7570 L=LL
7580 PP$=P$(1,L):P$=P$(L+1)
7590 LPRINT S$(1,LM);PP$;
7610 FOR J=1 TO LS:LIN=LIN+1:LPRINT ""
:NEXT J
7615 IF LIN>59 THEN FOR J=1 TO 66-LIN:
LPRINT "":NEXT J:LIN=0
7620 IF LEN(P$)>LL THEN L=LL:GOTO 7550
7630 IF CR AND LEN(P$)>0 THEN 7530
7640 RETURN
```

Subroutine to readjust lines in memory so that they fit properly on the screen after editing.

```
7999 REM TEXT REJUSTIFY RTN
8000 PRINT CL$:POSITION 2,5:PRINT "Re-
justifying text. . ."
8010 TL=LEN(T$):N=EL+NL-1:C=LP(N)+1:SL
=C-1:CH=C-LP(N-1)
8020 C$=T$(C,C)
8030 IF C$=CR$ THEN 8100
8035 IF C=TL THEN FL=1:SLOC=SL-LP(N-1)
:GOTO 8100
8040 IF C$=" " THEN SL=C
8050 IF CH<LWID-1 THEN CH=CH+1:C=C+1:G
OTO 8020
8060 IF SL=LP(N-1) THEN SL=C
8070 LP(N)=SL:C=SL+1:N=N+1:CH=1:GOTO 8
020
8100 LP(N)=C
8110 IF LP(N)=LP(N+1) THEN LN=LN-1:FOR
 I=N TO LN-1:LP(I)=LP(I+1):NEXT I
8120 RETURN
```

Subroutine to edit lines of text.

```
8999 REM EDIT SUBR
9000 FL=0:IF CHAR>1 THEN LP(LN)=LEN(T$
):LN=LN+1:LP(LN)=LEN(T$):L$="":FL=1
9005 POKE 752,1:IT=I:IF I>21 THEN V1=2
2:GOTO 9040
9010 V1=I+1:POSITION 2,VV
9020 X=21:IF X>LN-1 THEN X=LN-1
9025 IF X=IT THEN 9040
9030 FOR I=IT+1 TO X:PRINT T$(LP(I-1)+
1,LP(I)):NEXT I
9040 EL=V1+(IT>21)*(IT-21)-1
9050 Q$="UP/DN:Move RTN:Edit  D,X:Del
ESC:Exit"
9060 POSITION 2,0:PRINT Q$:PRINT LIN$
9080 C=ASC(T$(LP(EL-1)+1)):POSITION 2,
V1:PRINT CHR$(C+128);:GET #1,X
9085 POSITION 2,V1:PRINT CHR$(C);
9090 IF X<>45 THEN 9130:REM UP
9100 IF V1>2 THEN V1=V1-1:EL=EL-1:GOTO
 9080
9110 IF EL=1 THEN 9080
9115 EL=EL-5:IF EL<1 THEN EL=1
9118 NN=20:IF EL+NN>LN-1 THEN NN=LN-EL
-1
9120 POSITION 2,2:FOR I=EL TO EL+NN:?
S$;CHR$(31);T$(LP(I-1)+1,LP(I)):NEXT I
:GOTO 9080
9130 IF X<>61 THEN 9180:REM DOWN
9140 IF EL>=LN-1-FL THEN 9080
9150 EL=EL+1
9160 IF V1<22 THEN V1=V1+1:GOTO 9080
```

```
9165 NN=4:IF NN>LN-EL-1-FL THEN NN=LN-
EL-1-FL
9170 EL=EL+NN:POSITION 2,23:FOR I=EL-N
N TO EL:PRINT T$(LP(I-1)+1,LP(I)):NEXT
 I:GOTO 9060
9180 IF X=27 THEN 9580:REM ESC
9190 IF X<>68 OR V1=2 THEN 9250:REM D
9200 NC=LP(EL)-LP(EL-1):IF EL=LN-1 THE
N T$=T$(1,LP(EL-1)):GOTO 9205
9202 T$(LP(EL-1)+1)=T$(LP(EL)+1)
9205 FOR J=EL TO LN-1:LP(J)=LP(J+1)-NC
:NEXT J
9210 X=22-V1:IF X>LN-EL-2 THEN X=LN-EL
-2
9220 POSITION 2,V1:? S$:POSITION 2,V1:
IF EL<LN-1 THEN FOR J=EL TO EL+X:PRINT
 S$:? CHR$(28);T$(LP(J-1)+1,LP(J)):NEX
T J
9225 PRINT S$;
9230 IF EL=LN-1-FL THEN V1=V1-1:EL=EL-
1
9240 LN=LN-1:GOTO 9080
9250 IF X<>88 THEN 9310:REM X
9260 POSITION 2,0:PRINT "Delete from h
ere to the end of text?";:GET #1,X:IF
X<>89 THEN 9060
9270 LN=EL:L$="":CHAR=1:SLOC=0:IF LN>1
 THEN T$=T$(1,LP(LN-1)):GOTO 9280
9275 T$=""
9280 TL=LEN(T$):GOTO 9580
9310 IF X<>155 THEN 9080
9319 REM EDIT LINE
9320 L1=EL-8:IF L1<1 THEN L1=1
9330 L2=EL+8:IF L2>LN-1 THEN L2=LN-1
9340 PRINT CL$;"Type new line below (^
F to finish)  ":PRINT LIN$
9350 FOR J=L1 TO EL:PRINT T$(LP(J-1)+1
,LP(J)):NEXT J
9360 PRINT :PRINT :PRINT :PRINT
9370 IF L2>EL THEN FOR J=EL+1 TO L2:PR
INT T$(LP(J-1)+1,LP(J)):NEXT J
9380 POKE 752,0:POSITION 2,EL-L1+2
9390 TLN=LN
9410 C1=CHAR:S1=SLOC:CHAR=1:SLOC=0
9420 GOSUB 500
9430 CHAR=C1:SLOC=S1
9450 NL=LN-TLN:NC=LP(LN-1)-LP(TLN-1)
9480 IF EL=LN-1 THEN T$=T$(1,LP(EL-1))
:GOTO 9490
9485 T$(LP(EL-1)+1)=T$(LP(EL)+1)
9490 CC=LP(EL)-LP(EL-1):FOR J=EL TO LN
-1:LP(J)=LP(J+1)-CC:NEXT J:IF NL=0 THE
N LN=TLN-1:GOTO 9580
9500 X$=T$(LEN(T$)):IF X$<>CR$ AND X$<
>" " THEN T$(LEN(T$)+1)=" ":NC=NC+1:LP
(LN-2)=LP(LN-2)+1:LP(LN-1)=LP(LN-1)+1
9502 CN=LP(EL-1):FOR I=LEN(T$) TO CN+1
 STEP -NC:IF I<NC THEN T$(CN+NC+1,I+NC
)=T$(CN+1,I):GOTO 9504
9503 T$(I+1,I+NC)=T$(I-NC+1,I)
9504 NEXT I
9505 T$(CN+1,CN+NC)=T$(LEN(T$)-NC+1)
9506 FOR I=LN-2 TO EL STEP -1:LP(I+NL)
=LP(I)+NC:NEXT I
9508 J=TLN+NL-1:K=LP(LN-2)-LP(EL-1):FO
R I=0 TO NL-1:LP(EL+I)=LP(J+I)-K:NEXT
I
9510 T$=T$(1,LEN(T$)-NC)
9530 LN=TLN+NL-1:IF X$=CR$ OR EL=LN-NL
 THEN 9580
9550 SS=1:P$=T$(LP(EL+NL-1)+1,LP(EL+NL
)):LL=LEN(P$)
9560 IF P$(SS,SS)<>" " AND SS<LL THEN
SS=SS+1:GOTO 9560
9570 IF LP(EL+NL-1)-LP(EL+NL-2)+SS<=LW
ID THEN GOSUB 8000
9580 TL=LEN(T$):IF FL THEN LN=LN-1:L$=
T$(LP(LN-1)+1,LP(LN)):CHAR=LEN(L$)+1:T
L=LP(LN-1)
9600 STP=0:POKE 752,0:RETURN
```

Error-handling routine.

```
19999 REM ERROR-HANDLING RTN
20000 CLOSE #2:E=PEEK(195)
20010 POSITION 2,0:PRINT S$;:POSITION
2,0:PRINT "Error: Code ";E;"; press an
y key";
20020 GET #1,X
20030 TRAP 20000
20040 GOTO 200
```

SWAT

Strategic Weapon Against Typos

by Jon R. Voskuil
Atari version by Alan J. Zett

***SWAT (Strategic Weapon Against Typos)* is a debugging utility for any Atari.**

One of the major frustrations of typing computer programs from printed listings is finding typographical errors. The process is a time-consuming lesson in the computerist's version of Murphy's Law, "There is always one more bug."

Enter *SWAT* to the rescue! Inspired by a program with a similar aim that appeared in *Nibble* magazine, we've developed this program to help you find differences between the program listings in ***The Best of SoftSide*** and the lines you type into your computer.

Following this explanatory article, you will find *SWAT* Tables for each program in this book. These tables consist of columns of numbers and alphabetic codes, and are the result of running *SWAT*. The idea is that the tables in the book ought to be identical to the ones you generate on your computer. If there are any differences, then you know that the program in your computer's memory differs from the published program, and the table can tell you, within a few program lines, where to find the error.

What *SWAT* does

Swat generates three columns of information. Each entry in the first column is a range of line numbers; each entry in the second column is a two-letter *SWAT* Code; and each entry in the third column is the length in bytes (characters), of the specified program lines.

Superfluous or omitted characters will appear as differences in the third column. Mistyped characters will appear as differences in the second column.

How to use *SWAT*

First, type the listing of *SWAT* that appears immediately after this article, then LIST it to disk with the command LIST "D:SWAT", or LIST "C:" for cassette. Then, to use it on another program, you must append it to the end of the program to be checked. Here's how.

Type the program you want to test, then LIST it to tape or disk. Type "POKE 580,1", then press [SYSTEM RESET]. This will cause your Atari to re-boot, clearing the variable table. Alternatively, you may turn your Atari off and back on. Failure to re-boot may cause improper *SWAT* codes to be generated. After re-booting, load your program back into memory with the ENTER "C:" or ENTER "D:filename" command. Then append *SWAT* to your program with ENTER "C:" or ENTER "D:SWAT".

Once you have appended *SWAT* to your program, type GOTO 32000. You will have the opportunity to choose alternate "parameters" (more on them later), and whether to send the output to the screen or the printer. Once the *SWAT* Table has been generated, simply compare it to the published one. If they are identical, you may begin using your program.

What if the Tables Don't Match

First, examine the listed line numbers in the first columns of the *SWAT* Tables. If they don't match, it probably means that you have inserted, omitted, or changed one line or more. A gross error in the length of a line may also cause this type of discrepancy. An inserted or omitted line will affect all entries in the first column from that point on. Search the lines indicated by the earliest erroneous entry, and correct mistyped lines. Then repeat the *SWAT* procedure above.

If, after doing this, there are still discrepancies in the second or third columns, more deatiled trouble-shooting procedures will be necessary. A bad entry in column three will almost always be accompanied by a bad entry in column two, although the reverse may not be true.

If the length entry in column three is OK, but the corresponding *SWAT* code in column two is bad, the most likely cause is a simple substitution of one character for another somewhere within the indicated lines. For example, a variable name N0 may have been typed as NO; a comma may have been put in the place of a period; a number like 32767 may have been mistyped as 32757; or perhaps a word in a PRINT or REM statement may have been misspelled. There are more complicated possibilities, with the same number of bytes added in one part of a line as were omitted elsewhere in it. Also, keep in mind that BASIC keywords (PRINT, INPUT, FOR) occupy only one byte in memory. Thus, although GOSUB looks longer that GOTO, typing one for the other would change the *SWAT* code, but not the length.

If the entry in the third column is bad, it's possible to get some clue about the nature of the error by comparing the actual number to the published one. A number that is too large usually indicates extra characters, while a number that is too small usually indicates an omission. Remember that keywords may be deceptive in this context. The only way to find the typing errors is simply to compare what you typed line by line and character by character with the printed listing. *SWAT* narrows down the range you must search to no more that 12 lines or approximately 500 bytes of code. This "resolution" can easily be changed by answering "N" when *SWAT* asks you if you want to use standard parameters. You then enter, at the next prompt, the numbers following "NU=" and "B=", separated with a comma. If there is no indication in the published *SWAT* Table of modified parameters, just accept the standard ones.

Remember that *SWAT* is very picky, and "knows" nothing about how BASIC works. It may therefore balk at an insignificant difference in a REM, DATA, or PRINT statement. The only way to get correct *SWAT* Tables is to type *exactly* what appears in the printed listing.

Using *SWAT* to Check Itself

After typing *SWAT* and LISTing it to tape or disk, re-boot your Atari, ENTER *SWAT*, and add the following line:

32767 REM

Then, change the value of LN in line 32000 to 32767. You may then RUN the program, and it should generate the table printed right after the listing of *SWAT*.

```
  SS SS SS SS SS SS SS SS SS SS SS
  SS                            SS
  SS        Atari  BASIC        SS
  SS           'SWAT'           SS
  SS  Author:  Jon R. Voskuil   SS
  SS  Translator: Alan J. Zett  SS
  SS     Copyright (c) 1982     SS
  SS SoftSide Publications, Inc SS
  SS                            SS
  SS SS SS SS SS SS SS SS SS SS SS

32000 CLR :LN=32000:NU=12:B=500:DIM T$
(40)
32010 CLOSE #1:CLOSE #2:A=PEEK(136)+PE
EK(137)*256:GRAPHICS 0:OPEN #2,4,0,"K:
":DIM Z$(12):Z$="            "
32015 POKE 752,1:PRINT "}":PRINT :PRIN
T "NORMAL PARAMETERS? ":GET #2,P:PRINT
 :PRINT
32016 IF P<>89 THEN PRINT "INPUT # OF
LINES, # OF BYTES: ";:INPUT NU,B
32020 C1=1:C2=2:C3=3:CQ=256:N=0:PRINT
:PRINT :? "OUTPUT TO SCREEN OR PRINTER
? (S/P)":GET #C2,X
32022 TRAP 32020:CLOSE #C1:OPEN #C1,8,
0,CHR$(X):POKE 752,C1
32025 IF X=80 THEN ? :? "PROGRAM TITLE
: ";:INPUT T$:? #C1;CHR$(15):? #C1;" A
TARI SWAT TABLE FOR: ";T$:? #C1
32030 IF P<>89 THEN ? #C1;" (MODIFIED
PARAMETERS: NU=";NU;", B=";B;")":? #C1
32035 POKE 752,C1:? #C1;"    LINES
    SWAT CODE  LENGTH"
32040 ? #C1;" ------------   --------
-   ------"
32050 L1=PEEK(A)+PEEK(A+C1)*CQ:IF L1=L
N THEN CLOSE #C1:CLOSE #C2:POKE 752,0:
END
32060 S=0:A1=A:L=L1
32070 FOR I=C1 TO NU
32080 AA=A:A=A+PEEK(A+2)
32090 L2=L:L=PEEK(A)+PEEK(A+1)*CQ
32100 FOR J=AA TO A-C1:S=S+PEEK(J):NEX
T J
32110 IF L=LN OR A-A1>B THEN I=NU
32120 NEXT I
32130 D=INT(S/676):S=S-D*676:D1=INT(S/
26):D2=S-D1*26
32140 ? #1;Z$(1,6-LEN(STR$(L1)));L1;"
- ";L2;Z$(1,12-LEN(STR$(L2)));CHR$(D1+
65);CHR$(D2+65);
32150 ? #1;Z$(1,11-LEN(STR$(A-A1)));A-
A1:IF X<>83 THEN 32170
32160 N=N+1:IF N=19 THEN N=1:? #1:? #1
;"RETURN TO CONTINUE":GET #2,D:? #1
32170 GOTO 32050
```

SWAT TABLES

ATARI® SWAT TABLE FOR: **MELODY DICE**

LINES	SWAT CODE	LENGTH
5 - 90	JY	335
100 - 190	SE	375
200 - 1140	QD	638
1142 - 2070	NO	566
2080 - 2540	WQ	510
2542 - 2610	RF	546
2612 - 2650	IX	537
2652 - 2900	WY	355
3000 - 3050	SH	508
3060 - 3170	RZ	439
3180 - 3290	VF	358
3300 - 3410	NC	498
3420 - 3530	NZ	251
3540 - 3650	BC	330
3660 - 3900	CQ	416
3910 - 4090	BT	204
4100 - 4310	AR	255
4320 - 4640	TT	379
4650 - 4960	XG	519
4970 - 5100	DP	288
5110 - 5250	NV	251
5260 - 6050	LR	485
6060 - 6180	BB	271
6190 - 8000	SN	536
10000 - 10060	GE	534
10070 - 10150	PT	555
10160 - 10220	PK	547
10230 - 10300	IG	524

ATARI® SWAT TABLE FOR: **DATABASE II**

LINES	SWAT CODE	LENGTH
110 - 133	KY	512
134 - 138	RS	525
139 - 220	ZO	514
230 - 320	BP	527
330 - 440	UV	318
450 - 560	RY	354
570 - 680	PS	504
690 - 800	KW	479
810 - 910	JF	588
915 - 1000	OV	525
1010 - 1090	BW	554
1100 - 1210	JM	469
1220 - 1330	RB	465
1340 - 1450	TG	422
1460 - 1570	XA	438
1580 - 1690	JH	376
1700 - 1810	BF	529
1820 - 1930	UT	458
1940 - 2050	XE	416
2160 - 2270	LV	506
2280 - 2390	TQ	354
2400 - 2510	DO	450
2520 - 2630	SX	339
2640 - 2750	XE	219
2760 - 2870	EJ	432
2880 - 2980	MJ	534
2990 - 3100	KO	460
3110 - 3200	HF	441

ATARI® SWAT TABLE FOR: DEFENSE

LINES	SWAT CODE	LENGTH
10 - 40	CM	707
50 - 150	NH	539
155 - 230	NV	611
240 - 1000	MA	407
1005 - 3015	HU	536
3020 - 3045	IF	563
3050 - 3065	QP	527
3070 - 4040	LB	585
4050 - 4125	KJ	561
4130 - 6030	EB	533
6040 - 7000	SQ	618
7005 - 10045	DW	531
10050 - 11020	QA	507
11030 - 11120	FW	423

ATARI® SWAT TABLE FOR: SOLITAIRE

LINES	SWAT CODE	LENGTH
1 - 26	EQ	524
27 - 38	MR	577
40 - 50	FJ	542
52 - 110	CK	503
120 - 240	IT	472
250 - 360	MU	468
365 - 460	ZS	395
470 - 560	MH	566
570 - 680	US	365
690 - 785	EC	548
790 - 880	SK	495
890 - 980	XI	391
990 - 1100	XC	368
1110 - 1160	EF	519
1170 - 1240	CI	554
1250 - 1330	KB	529
1340 - 1450	FM	369
1460 - 1550	CV	622
1560 - 1630	BS	552
1640 - 1640	KM	16

ATARI® SWAT TABLE FOR: MAZE SEARCH

LINES	SWAT CODE	LENGTH
1 - 200	PO	296
210 - 254	BV	398
300 - 430	WV	541
500 - 600	OY	504
700 - 906	FF	284

ATARI® SWAT TABLE FOR: MUSIC PROGRAMMER

LINES	SWAT CODE	LENGTH
10 - 30	OR	532
31 - 42	KQ	624
45 - 82	GM	538
83 - 704	YG	505
705 - 720	XV	527
722 - 754	UC	551
755 - 806	NT	558
807 - 814	OU	510
816 - 826	OC	557
827 - 836	MR	564
837 - 910	UC	569
920 - 992	YM	521
993 - 1000	GR	579
1001 - 1017	XN	509
1019 - 1033	WU	515
1035 - 1085	WH	512
1090 - 1124	WE	527
1125 - 1150	GC	541
1151 - 1163	DA	514
1164 - 1185	RO	512
1186 - 1208	JO	528
1210 - 1222	NW	553
1224 - 1901	EG	547
1940 - 5016	WR	614
5017 - 6502	AS	512
6505 - 7530	YN	446
7532 - 7560	SR	547
7570 - 8453	AB	537
8454 - 9031	EF	605
9450 - 9469	IP	326

ATARI® SWAT TABLE FOR: **OPERATION SABOTAGE**

(Modified Parameters: NU = 5, B = 200)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
1 - 7	WH	176	1420 - 1450	BW	259
8 - 30	NO	174	1460 - 1490	UL	217
40 - 70	VW	241	1492 - 1502	TY	231
80 - 100	NK	229	1504 - 1520	SZ	153
110 - 130	YN	241	1522 - 1532	AI	216
140 - 160	RM	249	1534 - 1570	VN	171
170 - 200	SC	291	1580 - 1602	CY	245
202 - 230	UE	283	1604 - 1630	BK	225
240 - 270	RE	303	1640 - 1670	BN	242
272 - 300	FI	207	1680 - 1700	NI	231
310 - 320	NY	222	1710 - 1740	NN	211
330 - 350	QX	217	1750 - 1790	XF	267
360 - 380	NS	229	1800 - 1840	DK	277
390 - 402	TY	220	1850 - 1890	PB	174
410 - 430	WJ	222	1900 - 1932	RZ	201
440 - 470	KR	217	1940 - 1952	ZM	201
480 - 520	ZB	244	1960 - 1972	QA	268
530 - 570	BO	261	1980 - 1992	KU	218
580 - 650	NO	206	2000 - 2040	LN	148
660 - 700	BQ	168	2050 - 2080	HA	251
710 - 732	SX	174	2090 - 2100	QS	222
740 - 760	QE	209	2110 - 2132	CL	260
762 - 780	DU	216	2140 - 2154	AA	204
790 - 830	VN	138	2160 - 2200	RL	237
840 - 880	NL	299	2210 - 2250	QO	96
890 - 930	FV	101	2260 - 2300	AW	166
940 - 980	MX	105	2310 - 2342	BM	216
990 - 1030	RV	213	2344 - 2410	FZ	205
1040 - 1080	PH	113	2420 - 2460	ZQ	223
1090 - 1130	TJ	186	2470 - 2500	LS	228
1132 - 1150	BT	237	2510 - 2550	PG	178
1152 - 1170	UO	212	2560 - 2574	XK	221
1180 - 1210	EO	261	2580 - 2600	JX	201
1220 - 1242	SW	218	2610 - 2626	SK	374
1250 - 1270	SC	235	2630 - 2670	BG	201
1280 - 1310	DB	216	2700 - 2740	VB	217
1320 - 1360	NH	209	2742 - 2772	RE	275
1370 - 1410	UB	269	2774 - 2790	EL	185

ATARI® SWAT TABLE FOR:
THE FLIGHT OF THE BUMBLEBEE BEE
(Modified Parameters: NU=5, B=200)

LINES	SWAT CODE	LENGTH
10 - 40	CO	264
100 - 120	CA	205
130 - 140	XG	244
150 - 170	EC	233
180 - 190	LO	250
200 - 1000	DT	210
1001 - 1005	NK	169
1010 - 1060	MC	204
1080 - 1140	HT	259
1160 - 1220	RU	213
1240 - 1280	SZ	206
1300 - 1340	CP	206
1360 - 1420	QC	233
1440 - 1500	GK	220
1520 - 1580	ZR	212
1600 - 1660	XT	212
1680 - 1740	HW	212
1760 - 1820	EP	212
1840 - 1900	NW	234
1920 - 1980	SQ	229
19999 - 30200	BN	344
30210 - 30310	LN	279
30320 - 30330	MD	101

ATARI® SWAT TABLE FOR:
FUGUE
(Modified Parameters: NU=5, B=200)

LINES	SWAT CODE	LENGTH
1 - 20	GU	217
210 - 230	AL	215
240 - 300	NQ	202
310 - 330	AR	243
1000 - 1040	FB	205
1050 - 1090	ZO	213
1100 - 1140	BY	209
1150 - 1190	HL	207
1200 - 1240	SH	231
1250 - 1290	WV	239
1300 - 1340	UU	234
1350 - 1390	AO	232
1400 - 1440	SQ	225
1450 - 1480	KK	202
1490 - 1530	IB	249
1540 - 1570	FD	212
1580 - 1610	HY	209
1620 - 1650	XM	215
1660 - 1690	UN	221
1700 - 1730	PM	204
1740 - 1770	UC	215
1780 - 1810	XS	224
1820 - 1850	KO	225
1860 - 1870	KY	114

ATARI® SWAT TABLE FOR:
MINIGOLF

LINES	SWAT CODE	LENGTH
8 - 38	RP	538
40 - 74	AN	519
90 - 108	VS	510
109 - 152	UC	542
154 - 205	QS	537
210 - 231	LM	517
232 - 310	AJ	554
320 - 420	ZQ	515
430 - 510	XP	542
512 - 600	AZ	537
602 - 630	GQ	510
640 - 700	OT	533
702 - 750	JI	571
752 - 804	IZ	540
810 - 852	YX	519
854 - 910	HX	576
915 - 1000	ON	607
1002 - 1020	BZ	501
1030 - 1100	AM	547
1102 - 1150	DQ	524
1152 - 2030	NJ	501
2040 - 9999	PJ	209

ATARI® SWAT TABLE FOR: **FLIP-IT**

LINES	SWAT CODE	LENGTH
100 - 160	IV	565
170 - 200	SV	556
205 - 330	XH	501
340 - 632	SN	431
640 - 706	WC	542
708 - 832	QI	513
840 - 1000	NN	546
1005 - 1040	DV	530
1045 - 1080	JB	554
1085 - 1120	TO	578
1125 - 1160	EY	545
1165 - 2008	XD	579
2010 - 2050	WQ	552
2060 - 3032	TB	534
3034 - 3080	CT	333
3082 - 3130	XC	454
3132 - 4020	IP	489
4050 - 4092	CQ	518
4100 - 4132	EY	535
4134 - 4174	NB	530
4180 - 4230	GN	596
4240 - 5040	NB	564
5050 - 5100	MN	584
5110 - 7010	SV	556
7020 - 7030	KK	71

ATARI® SWAT TABLE FOR: **WORD SEARCH PUZZLE GENERATOR**

LINES	SWAT CODE	LENGTH
90 - 135	BT	529
140 - 210	IK	502
220 - 330	DZ	457
340 - 450	FJ	413
460 - 570	GI	462
580 - 690	AQ	347
692 - 760	PG	290
765 - 770	NQ	56

ATARI® SWAT TABLE FOR: **BATTLEFIELD**

LINES	SWAT CODE	LENGTH
10 - 40	AF	507 ✓ ✓
42 - 130	WO	258 ✓ ✓
140 - 1080	IK	533 ✓ ✓
1090 - 2050	MO	509 ✓ ✓
2055 - 2170	TP	470 ✓ ✓
2175 - 3060	WB	439 ✓ ✓
4000 - 4115	FN ⊖ -	212 ⊖ ✓
4130 - 5060	SA	257 ✓ ✓
5070 - 6030	HI -	309 ✓
6035 - 6150	WA - -	275
6155 - 7080	GB - -	344
7090 - 8020	NC - -	344
8025 - 8110	FH - -	496
8120 - 10110	IU - -	329
10120 - 11060	CM - -	302
11070 - 12080	XT	408 ✓ ✓
12090 - 13070	DC - -	412 ⊖ ⊖ -
13080 - 13120	JW -	58 ✓

ATARI® SWAT TABLE FOR: **CHARACTER GENERATOR**

LINES	SWAT CODE	LENGTH
10 - 36	HF	400
38 - 72	OM	414
73 - 102	MQ	306
104 - 172	YR	455
174 - 280	EM	401
290 - 358	UJ	478
360 - 470	AM	327
480 - 566	GZ	413
568 - 630	FK	370
640 - 690	JY	528
700 - 810	IC	495
820 - 910	XM	416
920 - 1030	YZ	364
1040 - 10050	FH	402
10051 - 10070	CI	138

ATARI® SWAT TABLE FOR: GAMBLER

LINES	SWAT CODE	LENGTH
1 - 20	TU	534
22 - 80	OF	552
81 - 100	WZ	503
110 - 156	TO	516
160 - 200	WI	527
202 - 250	XT	502
252 - 310	ZD	583
320 - 361	PJ	632
362 - 390	OJ	531
392 - 421	EO	563
450 - 473	MG	512
480 - 522	VZ	570
524 - 542	KT	557
544 - 600	XF	534
602 - 631	AC	530
632 - 662	ZE	527
664 - 714	JH	670
716 - 734	EB	549
735 - 750	TW	569
751 - 770	OT	539
780 - 810	JD	545
820 - 870	QY	529
880 - 930	GA	519
932 - 960	SL	544
962 - 1002	JJ	584
1004 - 1030	TJ	507
1032 - 1070	IF	648
1072 - 1122	EV	567
1130 - 1172	SW	572
1174 - 1202	JK	589
1204 - 1234	EY	538
1240 - 1280	XW	581
1282 - 1320	BY	556
1330 - 1362	XL	608
1370 - 1450	VA	522
1452 - 1475	AP	580
1480 - 1482	VM	105

ATARI® SWAT TABLE FOR: LEYTE

LINES	SWAT CODE	LENGTH
50 - 100	OA	534
110 - 127	KL	566
130 - 176	DC	518
177 - 240	YE	537
250 - 320	ID	510
330 - 420	UL	555
425 - 510	UY	526
520 - 560	CH	555
570 - 610	VE	550
615 - 680	ZA	597
690 - 770	NN	519
780 - 885	TQ	312
890 - 940	HG	548
950 - 980	KL	511
1000 - 1040	OL	514
1050 - 1100	GZ	511
1105 - 2080	KD	550
2090 - 3060	AZ	574
3070 - 4040	OE	535
4050 - 6030	QF	529
6040 - 6110	AI	503
6120 - 7010	SD	164

ATARI® SWAT TABLE FOR: SPACE RESCUE

LINES	SWAT CODE	LENGTH
10 - 40	WU	531
45 - 80	MM	516
81 - 147	BR	503
150 - 220	WI	522
225 - 285	XI	546
290 - 400	RW	550
410 - 465	ZU	528
470 - 550	GP	554
560 - 670	JO	575
680 - 699	ZV	546
1000 - 1060	TB	530
1070 - 1140	BX	555
1142 - 1152	TO	642
1154 - 1160	HU	300

APPLE™ SWAT TABLE FOR: **QUEST** (Modified Parameters: NU = 5, B = 200)

LINES	SWAT CODE	LENGTH
1 - 7	JO	290 ✓
8 - 100	UO	207 ✓
102 - 108	EF –	228 –
110 - 125	VM –	167
130 - 150	ZT	108 ✓
155 - 175	JL	111 ✓
180 - 200	ZG	116 ✓
205 - 225	HO	118 ✓
240 - 280	LR	117 ✓
290 - 330	XA	115 ✓
340 - 380	NZ	118 ✓
390 - 430	XK	118 ✓
440 - 480	HG	118 ✓
490 - 530	YI	118 ✓
540 - 580	YD	115 ✓
590 - 602	QT	246 ✓
603 - 610	QM	269 ✓
615 - 810	HS	297 ✓
812 - 830	OI	216 ✓
832 - 840	OC	216 ✓
845 - 860	EU	228 ✓
862 - 880	RL	226 ✓
882 - 903	VA	266 ✓
905 - 911	YV –	204
912 - 916	JK	131 ✓
917 - 925	TU	226 ✓
930 - 942	EG –	255
945 - 955	JM	234 ✓
957 - 980	XI	113 ✓
990 - 994	OD –	268 –
995 - 998	TE –	292 –
999 - 1003	EF –	200
1004 - 1020	UM –	191 –
1022 - 1030	TF –	169
1031 - 1060	QL –	239
1061 - 1065	JY	271 ✓
1070 - 1077	IV	206 ✓
1080 - 1086	YJ	218 ✓
1087 - 1088	NK	246 ✓
1090 - 1097	XU –	178
1100 - 1120 –	ZZ –	140 –
1125 - 1145	RE	198 ✓
1200 - 1207	MP –	178
1210 - 1230	RD –	171
1235 - 5000	SZ –	173
5005 - 5020	BS	263 ✓
5025 - 10015	CV	212 ✓
10020 - 10030	WG –	224 –
10032 - 10042	JP	213 ✓
10045 - 11014	LF	256 ✓
11015 - 11030	VY	244 ✓
11032 - 11042	UO	211 ✓
11045 - 15100	JG –	190
15102 - 15200	LO –	167
15202 - 15300	DK –	163
15302 - 15320	KK –	195
15400 - 15410	PW –	205
15420 - 15507	DY –	128
15510 - 15522	RJ	123 ✓
15523 - 15601	BG –	83
15605 - 15610	ZD	135 ✓
15620 - 15699	NO –	144
15700 - 15705	YV –	155
15706 - 15710	RJ –	123
15711 - 15720	QT –	200
15750 - 16002	WR –	143
16003 - 16008	YV	147 ✓
16010 - 16020	UB –	120
17000 - 17015	ZO –	176
17020 - 17024	GL	201 ✓
17025 - 18005	PR –	188
20000 - 20010	SV	236 ✓
20011 - 20019	WA –	234
20020 - 20025	WI	243 ✓
20027 - 20035	VU –	222
20036 - 20050	MV	137 ✓
20055 - 20099	LO	228 ✓
21000 - 21005	YW –	317 –
21010 - 21022	HN –	185
21025 - 30000	JL –	257 –
30010 - 30021	PA –	228 –
30030 - 30050	LD –	153 –

ATARI® SWAT TABLE FOR: **TITAN PART 1**

LINES	SWAT CODE	LENGTH
1100 - 1140	RL	589
1150 - 1200	NZ	509
1210 - 1250	JO	527
1260 - 2160	PW	532
2170 - 3020	GL	569
3030 - 20020	VY	593
20030 - 20070	ZA	507

ATARI® SWAT TABLE FOR: **TITAN PART 2**

LINES	SWAT CODE	LENGTH
300 - 1030	NE	574
1040 - 1140	LV	516
1150 - 1240	VC	589
1250 - 1310	DL	553
1320 - 1350	CZ	540
1360 - 2020	RB	606
2030 - 2120	AZ	561
2130 - 2185	UA	560
2190 - 2250	ZM	518
2300 - 2350	JQ	549
2360 - 2445	HL	529
2450 - 2500	LW	504
2510 - 2550	WP	533
2560 - 2650	GP	524
2660 - 3010	RZ	532
3020 - 3500	FS	535
3510 - 3550	AP	543
3560 - 4105	LU	579
4110 - 4150	ZP	603
4160 - 5000	VO	566
5010 - 6000	SP	517
6010 - 8000	SQ	505
8010 - 8550	BH	532
8560 - 8620	RU	585
8630 - 20020	PM	582
20030 - 20070	MK	507

ATARI® SWAT TABLE FOR: **MICROTEXT 1.2**

LINES	SWAT CODE	LENGTH
10 - 130	UG	528
135 - 499	WS	380
500 - 720	EU	239
739 - 879	TI	222
880 - 1110	ZV	328
1115 - 2085	TW	284
2090 - 3040	VQ	378
3050 - 3180	OA	390
3190 - 4075	ZN	323
4080 - 4240	BP	500
4250 - 5082	PC	361
5083 - 5250	SE	430
5255 - 6999	IG	246
7000 - 7100	FF	539
7110 - 7550	XK	268
7560 - 8010	IO	419
8020 - 9000	YX	435
9005 - 9100	AY	500
9110 - 9190	CG	514
9200 - 9260	QO	526
9270 - 9380	CK	460
9390 - 9502	DK	549
9503 - 9580	MB	542
9600 - 20040	HX	213
60000 - 60110	TM	449
60120 - 60130	HG	110

ATARI® SWAT TABLE FOR: **SWAT**

LINES	SWAT CODE	LENGTH
32000 - 32025	JU	581
32030 - 32130	DZ	455
32140 - 32170	KP	293

Typing Got You Down?

The Best of SoftSide contains over twenty programs for your Atari® — that's one big typing job. You'll spend hours at the keyboard, and face the inescapable fact of typographical errors.

Wouldn't you like to:

- Avoid typing?
- Use and enjoy these programs right away?
- Avoid worry about typos?

That's why we've prepared *The Best of SoftSide* Disk Version. That's right — on four diskette sides, every single program in *The Best of Soft Side.* You just insert the disk into your disk drive, and leave the typing to us. All you have to do is use and enjoy *The Best of SoftSide.*

If the form below is missing, send your name and address, along with $49 (check or money order), to:

***SoftSide* Publications, Inc.,**
Dept. BOSDV-AT, 6 South Street, Milford, NH 03055.

Order with this form or facsimile.

Please send me the Atari® *Best of SoftSide* on Disk.

Credit card holders may call (603) 673-0585, and have your VISA or MasterCard ready.

Name ______________________________

Address ______________________________

City ______________________ State ______ Zip ______

$49 to be paid by: ☐ Check ☐ M.O. ☐ VISA ☐ MasterCard

Name of Cardholder ______________________________

MC# and Interbank#/VISA# ______________________________

Exp. Date ______________________________

Signature ______________________________

Allow 4-6 weeks for delivery.

Price subject to change without notice. Atari® is the registered trademark of Warner Communications.

SoftSide™

READ THIS:

If the high price of commercial software and the lack of clear information about your microcomputer has got you down, here's the solution you've been waiting for!

SoftSide Magazine

SoftSide is a favorite of computer users and hobbyists alike. They rely on it as a prime source of programs, reviews and articles for the Apple®, Atari®, TRS-80®, and IBM® PC microcomputers. **SoftSide** is the magazine for the microcomputer owner who wants to learn BASIC programming, learn MORE about BASIC programming, or just wants to have FUN! **SoftSide** gives you the BASIC code listings of several programs — adventures, utilities, games, simulations, you name it — for your computer EVERY MONTH.

There's more:

- **Reviews** — of the software and hardware products you want to know about.
- **Articles** — about all aspects of BASIC programming.
- **Tutorials** — on graphics, use of important commands, and more.
- **Programs** — each month **SoftSide** publishes a variety of programs for the Apple®, Atari®, TRS-80®, and IBM® PC.

As you can see, you'll receive pages and pages of information and entertainment from SoftSide. Here's the best part:

A subscription to SoftSide is just $30 a year. That's 12 issues for only $2.50 each! What a value!

Use coupon to order. Mail to: **SoftSide Publications, Inc.**, 6 South St., Milford, NH 03055

☐ **YES! Send me the first copy of my SoftSide subscription right away!**

$30/yr for USA/Canada/APO & FPO only. For orders to Mexico — $46/yr. Other foreign orders — $68/yr.

I own an ☐ Apple® ☐ Atari® ☐ TRS-80® ☐ IBM® PC

☐ MasterCard ☐ VISA ☐ Check is enclosed

Name of Cardholder ____________

Name ____________

Address ____________

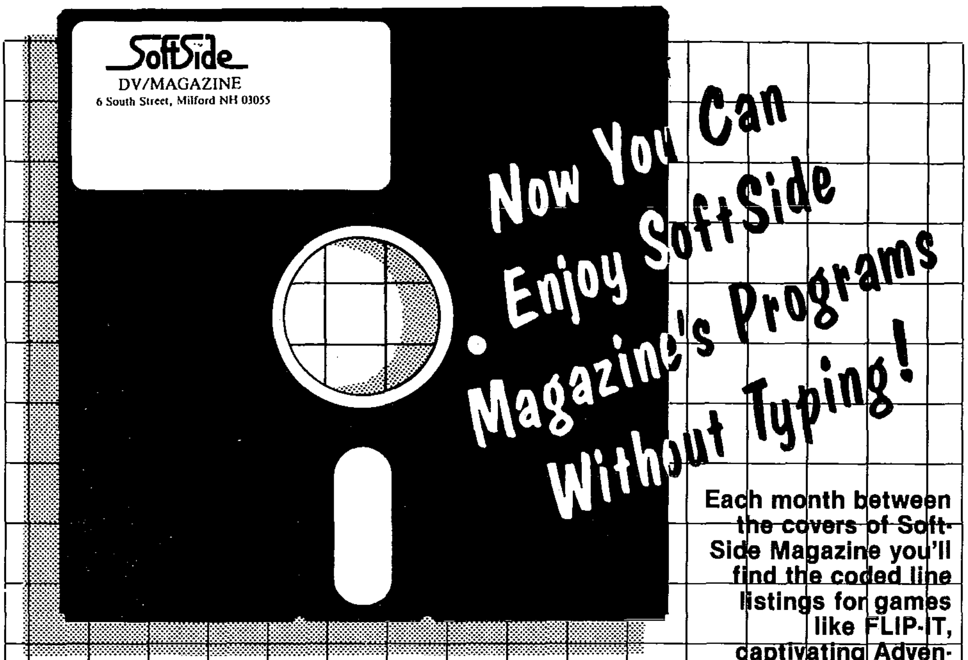
MC# and Interbank#/VISA# ____________

City/State ____________

____________ Exp. Date ____________

Zip ____________

Signature ____________



Each month between the covers of Soft-Side Magazine you'll find the coded line listings for games like FLIP-IT, captivating Adventures like QUEST, and valuable utilities like MICROTEXT, a complete word processor program.

But did you know that the same quality programs featured in SoftSide (the magazine), are now also available by annual subscription on disk (DV), and cassette (CV)? On the disk version, you also receive additional programs — two not published in Soft-Side Magazine.

So, if you want to enjoy quality programs at an economical price, subscribe to SoftSide Magazine. But if you want all that plus less time at your keyboard, order your disk version... or cassette version... of SoftSide, today!

Use This Coupon or Facsimile to Order

Make Check or M.O. payable to:

SoftSide
6 South Street
Milford, NH 03055

Name ______________________

Address ______________________

City ____________ State ______ Zip ______

Prices subject to change without notice. Apple®, IBM®, Atari®, and TRS-80® are registered trademarks of The Apple Computer Company, IBM, Inc. Warner Communications and The Tandy Corporation respectively.

I am sending:

☐ $149/yr for SoftSide Magazine Plus Disk Version

☐ $99/yr for SoftSide Magazine Plus Cassette Version

(Prices for USA Only)

Send me the ☐ Apple® ☐ Atari® ☐ TRS-80® Mod I ☐ TRS-80® Mod III ☐ IBM® PC (Disk Only) Version

I am paying by: ☐ Check or M.O. ☐ MasterCard ☐ VISA

Name of Cardholder ______________________

MC#/Interbank#/VISA# ______________________

______________________ Exp. Date __________

Signature ______________________

Disk Version Directory

Program Name	File Name
Defense	DEFENSE
Quest	QUEST
Quest2	QUEST2
Space Rescue	SPACERES
Maze Search	MAZESRCH
Minigolf	MINIGOLF
Flip-It	FLIPIT
Battlefield	BTLFLD
Solitaire	SOLITAIR
Gambler	GAMBLER
Operation Sabotage	OPSAB
Leyte	LEYTE
Titan	TITAN
Titan Part 2	TITAN 32K.PT2
Word Search Puzzle Generator	WDPUZGEN
Fugue	FUGUE
Flight of the Bumblebee	BMBLBEE
Melody Dice	MLDICE
Music Programmer	MUSIPROG
Character Generator	CHARGEN
Random Access Database	DATABASE
Microtext 1.2 Word Processor	MICROTXT
SWAT	SWAT

NOTES:

NOTES:

NOTES:

NOTES:

THE BEST OF SoftSide™

- Become a fearless fighter on a **Quest** through a labyrinthine dungeon!
- Become an admiral in the Battle of **Leyte**!
- Spend a leisurely afternoon playing **Solitaire**.
- Manage your information with **Data Base**.
- Write letters with **Microtext**.
- Much, much more...Utilities...Games... Adventures...

For over four years, **SoftSide,** "Your BASIC Software Magazine," has been bringing thousands of Apple®, Atari®, and TRS-80® users the best in BASIC entertainment software, as well as reviews and articles, all designed to help you get more out of your microcomputer.

Now, from **SoftSide's** past...we've selected the most useful...the most entertaining...the most fun-filled programs we've ever published. **The Best of SoftSide** offers you many hours of fun and programs of enduring value.

SOFTSIDE PUBLICATIONS, INC.
Milford, New Hampshire