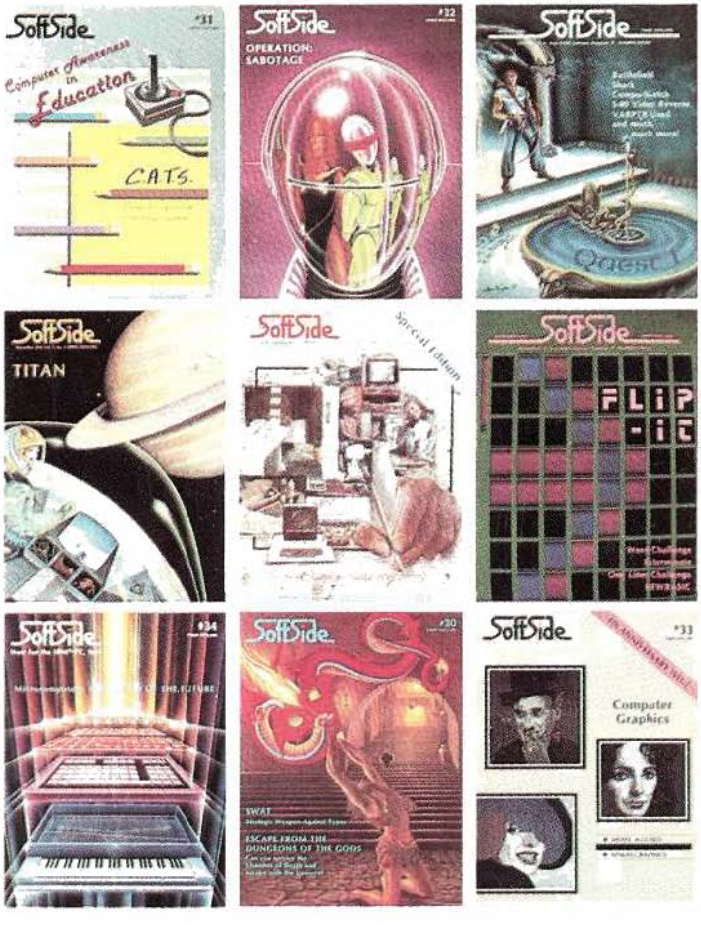
TRS-80® Edition

THE BEST OF SoftSide™



ADVENTURES • GAMES • UTILITIES

THE BEST OF

SoftSide™

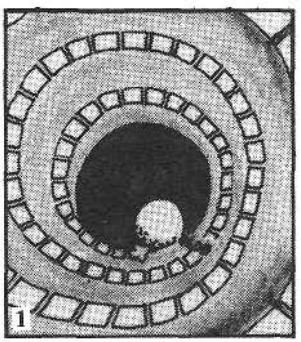
TRS-80® PROGRAMS for the TRS-80® Model I/III Microcomputers

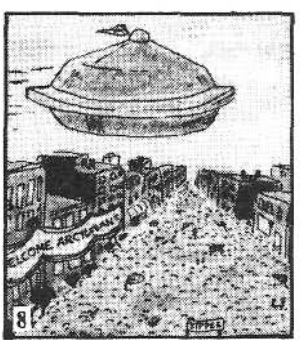
Edited by Fred Condo

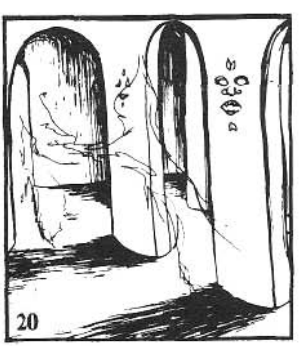
SOFTSIDE PUBLICATIONS, INC.
Milford, New Hampshire

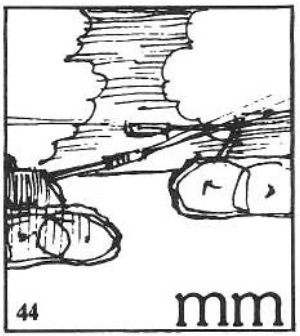
THE BEST OF SoftSide™

CONTENTS









Adventure

Simulations

Word Game

Graphics Utility

Practical Applications

Appendix

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

Each program in the book is introduced and explained by an article, followed by a listing of the BASIC program itself. These listings were generated by a TRS-80 like yours, in a 64-column format that matches what you should see on your screen. Except where specifically noted in the explanatory article, all the programs will run, unmodified, on both Model I and Model III TRS-80 computers.

We've also included *SWAT*, the Strategic Weapon Against Typos, which will prove to be a great aid to your typing. This debugging program appears last in the book, while the *SWAT* tables it generates immediately follow each program. The *SWAT* article explains how to use the program to locate and eliminate 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.***

Minigolf

by Mitch Voth

***Minigolf* is an arcade/simulation game for a TRS-80 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 TRS-80. *Minigolf* is a graphic representation of a nine-hole miniature golf course with banked walls and even some sound. For more fun, the program accommodates 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       TRS-80  BASIC         SS
SS         'Minigolf'          SS
SS    Author:  Mitch Voth      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 variables, and input the number of players and their names.

```
10 CLS:CLEAR200:DEFINTA-H:DEFSTRZ:PRINTCHR$(23):PRINT@18,"TRS-80
  9 HOLE";:PRINT@382,"MINIATURE GOLF"
30 PRINT@400,"NUMBER OF PLAYERS";:INPUTNP:IFNP>10,10ELSEPRINT@40
0,CHR$(30);:FORA=1TONP:PRINT@400,"PLAYER #";A;:INPUTZ(A):PRINT@4
00,CHR$(30);:NEXT:CLS:GOTO300
```

Score subroutine

```
40 CLS:PRINTCHR$(23);:PRINT@12,"SCORE:":FORA=1TONP:PRINTZ(A),S(A)
:NEXT:FORA=1TO4000:NEXT:CLS:RETURN
```

Input routine to get putter movement and numeric input for hitting ball.

```
50 Z=INKEY$:IFZ<>"",IFASC(Z)<58ANDASC(Z)>48,HH=VAL(Z)*10:POKEHP,
46:HM=0:VM=0:IFA=1,RESET(C1,C2):RESET(C1+1,C2):GOTO200ELSESET(C1
,C2):SET(C1+1,C2):GOTO200
60 T=0:IFZ<>"",IFASC(Z)=8,CP=CP+1:IFCP=17,CP=1:IFA=1,RESET(C1,C2
):RESET(C1+1,C2):GOTO100ELSESET(C1,C2):SET(C1+1,C2):GOTO100ELSEI
FA=1,RESET(C1,C2):RESET(C1+1,C2):GOTO100ELSESET(C1,C2):SET(C1+1,
C2):GOTO100
70 IFZ<>"",IFASC(Z)=9,CP=CP-1:IFCP=0,CP=16:IFA=1,RESET(C1,C2):RE
SET(C1+1,C2):GOTO100ELSESET(C1,C2):SET(C1+1,C2):GOTO100ELSEIFA=1
,RESET(C1,C2):RESET(C1+1,C2):GOTO100ELSESET(C1,C2):SET(C1+1,C2):
GOTO100
80 GOTO50
90 POKEHP,46:SET(B1,B2):SET(B1+1,B2):IFPOINT(C1,C2)=-1,A=0:RESET
(C1,C2):RESET(C1+1,C2):GOTO50ELSESET(C1,C2):SET(C1+1,C2):A=1:GOT
050
```

Routine to alter ball or putter position values.

```
100 ONCPGOTO101,102,103,104,105,106,107,108,109,110,111,112,113,
```

```
114,115,116
101 C1=B1:C2=B2+2:M1=0:M2=-1:GOTO90
102 C1=B1+2:M1=-1:GOTO90
103 C1=B1+4:C2=B2+2:M1=-2:M2=-1:GOTO90
104 C2=B2+1:M2=-.5:GOTO90
105 C2=B2:M2=0:GOTO90
106 C2=B2-1:M2=.5:GOTO90
107 C1=B1+4:C2=B2-2:M1=-2:M2=1:GOTO90
108 C1=B1+2:M1=-1:GOTO90
109 C1=B1:M1=0:GOTO90
110 C1=B1-2:M1=1:GOTO90
111 C1=B1-4:C2=B2-2:M1=2:M2=1:GOTO90
112 C2=B2-1:M2=.5:GOTO90
113 C2=B2:M2=0:GOTO90
114 C2=B2+1:M2=-.5:GOTO90
115 C1=B1-4:C2=B2+2:M1=2:M2=-1:GOTO90
116 C1=B1-2:M1=1:GOTO90
```

Hole-in-one routine.

```
150 IFS=1,PRINT@153,"A HOLE IN ONE!!";:FORA=1TO2000:NEXT:PRINT@1
53,CHR$(30);:RETURNELSEPRINT@149,"THAT TOOK YOU";S;"STROKES";:FO
RA=1TO2000:NEXT:PRINT@149,CHR$(30);:RETURN
200 D1=B1:D2=B2:RESET(B1,B2):RESET(B1+1,B2):IF(ABS(M1)=1ANDHM=1)
ORABS(M1)=2,D1=B1+2*SGN(M1):HM=0ELSEHM=1
205 T=T+1:IFT>4,RC=2:RETURN
210 IFPOINT(D1,B2)=-1,RC=0:RETURNELSESET(D1,B2):SET(D1+1,B2):B1=
D1:IFPEEK(HP)<>46,IFPEEK(HP)=140,S(PT)=S(PT)+S-P:RC=1:GOSUB150:R
ETURNELSEPOKEHP,46
220 RESET(B1,B2):RESET(B1+1,B2):IF(ABS(M2)=.5ANDVM=1)ORABS(M2)=1
,D2=B2+SGN(M2):VM=0ELSEVM=1
230 IFPOINT(B1,D2)=-1,RC=0:RETURNELSESET(B1,D2):SET(B1+1,D2):B2=
D2:IFPEEK(HP)<>46,IFPEEK(HP)=140,S(PT)=S(PT)+S-P:RC=1:GOSUB150:R
ETURNELSEPOKEHP,46
240 T=0:HH=HH-1:IFHH<0,RC=2:RETURNELSE200
```

Display score.

```
300 FORA=50TO79:SET(A,44):SET(A,9):NEXT:FORA=9TO44:SET(50,A):SET
(51,A):SET(78,A):SET(79,A):NEXT:HP=15648:PRINT@25,"HOLE # 1   PA
R 2";:P=2:FORPT=1TONP:S=1:PRINT@91,CHR$(30);:PRINT@91,Z(PT);"'S
TURN";:B1=RND(11)*2+52:B2=41:SET(B1,B2):SET(B1+1,B2):POKEHP,46:
CP=1
305 GOSUB100
310 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO310
```

```
320 IFRC=1,NEXTPT:GOTO400
325 HH=HH-10:IFHH<2,HH=2
330 IFD1=500RD1=78,M1=-M1:GOSUB200:GOTO310ELSEM2=-M2:GOSUB200:GO
TO310
```

Routines for displaying individual holes on screen.

```
400 GOSUB40:FORA=50TO79:SET(A,44):SET(A+22,9):NEXT:FORA=20TO43:S
ET(50,A):SET(51,A):SET(78,A):SET(79,A):NEXT:SET(78,19):SET(79,19
):FORA=78TO101:SET(A,18):NEXT:FORA=10TO17:SET(100,A):SET(101,A):
NEXT:FORA=52TO71:SET(A,45.5-A/2):NEXT:HP=15863
405 PRINT@25,"HOLE # 2   PAR 2";:P=2:FORPT=1TONP:S=1:PRINT@91,CH
R$(30);:PRINT@91,Z(PT);"'S TURN";:B1=RND(11)*2+52:B2=41:POKEHP,4
6:CP=1:SET(B1,B2):SET(B1+1,B2):GOSUB100
420 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO420
430 IFRC=1,NEXTPT:GOTO500
435 HH=HH-10:IFHH<2ANDHH>-6,HH=2
440 IFD2=90RD2=44OR(D2=18ANDD1>79),M2=-M2:GOSUB200:GOTO420ELSEIF
D1=500RD1=780RD1=100,M1=-M1:GOSUB200:GOTO420ELSEA=M1:M1=-M2*2:M2
=-A/2:GOSUB200:GOTO420
500 GOSUB40:FORA=20TO43:SET(22,A):SET(23,A):SET(50,A):SET(51,A):
SET(78,A):SET(79,A):NEXT:FORA=22TO79:SET(A,44):NEXT:FORA=44TO57:
SET(A,9):NEXT:FRINT@409,CHR$(191);:FORA=10TO19:SET(63-A*2,A):SET
(62-A*2,A):SET(38+A*2,A):SET(39+A*2,A):NEXT:HP=16210
510 PRINT@25,"HOLE # 3   PAR 2";:P=2:FORPT=1TONP:S=1:PRINT@91,CH
R$(30);:PRINT@91,Z(PT);"'S TURN";:B1=RND(11)*2+52:B2=41:SET(B1,B
2):SET(B1+1,B2):POKEHP,46:CP=1:GOSUB100
520 IFRC=2,S=S-1:CP=1:GOSUB100:GOTO520
530 IFRC=1,NEXTPT:GOTO600
540 HH=HH-10:IFHH<2ANDHH>-6,HH=2
550 IFD2=90RD2=44,M2=-M2:GOSUB200:GOTO520ELSEIFD1=220RD1=500RD1=
78,M1=-M1:GOSUB200:GOTO520ELSEIFD1<50,A=M1:M1=-M2*2:M2=-A/2:GOSU
B200:GOTO520ELSEA=M1:M1=M2*2:M2=A/2:GOSUB200:GOTO520
600 GOSUB40:FORA=17TO44:SET(50,A):SET(51,A):SET(78,A):SET(79,A):
NEXT:FORA=17TO29:SET(98,A):SET(99,A):SET(116,A):SET(117,A):NEXT:
FORA=66TO101:SET(A,9):NEXT:FORA=52TO77:SET(A,44):NEXT:FORA=94TO1
01:SET(A,37):NEXT:FORA=80TO97:SET(A,17):NEXT
610 FORA=10TO16:SET(85-2*A,A):SET(84-2*A,A):SET(83+A*2,A):SET(82
+A*2,A):SET(60+A*2,A+20):SET(61+A*2,A+20):SET(134-A*2,A+20):SET(
135-2*A,A+20):NEXT:HP=15852:PRINT@25,"HOLE # 4   PAR 3";:P=3:FOR
PT=1TONP:S=1:PRINT@91,CHR$(30);:PRINT@91,Z(PT);"'S TURN";
615 CP=1:B1=RND(11)*2+52:B2=41:SET(B1,B2):SET(B1+1,B2):GOSUB100
620 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO620
630 IFRC=1,NEXTPT:GOTO700
640 HH=HH-10:IFHH<2ANDHH>-6,HH=2
```

```
650 IFD2=90RD2=440R(D2=37ANDD1>90)OR(D2=17ANDD1>70ANDD1<100),M2=
-M2:GOSUB200:GOTO620ELSEIFD1=500RD1=780RD1=980RD1=116,M1=-M1:GOS
UB200:GOTO620ELSEIF(D1>80ANDD2<20)OR(D2>25ANDD1<96),A=M1:M1=M2*2
:M2=A/2:GOSUB200:GOTO620ELSEA=M1:M1=-M2*2:M2=-A/2
660 GOSUB200:GOTO620
700 GOSUB40:FORA=19TO44:SET(78,A):SET(79,A):NEXT:FORA=50TO77:SET
(A,44):NEXT:FORA=32TO43:SET(50,A):SET(51,A):NEXT:FORA=42TO49:SET
(A,32):NEXT:FORA=19TO22:SET(22,A):SET(23,A):NEXT:FORA=42TO59:SET
(A,9):NEXT:FORA=9TO18:SET(42+A*2,A):SET(43+A*2,A)
710 SET(59-A*2,A):SET(58-A*2,A):SET(A*2+4,A+14):SET(A*2+5,A+14):
SET(40+A*2,A+14):SET(41+A*2,A+14):NEXT:HP=15827:PRINT@25,"HOLE #
 5   PAR 2";:P=2:FORPT=1TONP:S=1:PRINT@91,CHR$(30);:PRINT@91,Z(P
T);"'S TURN";:B1=RND(11)*2+52:B2=41:SET(B1,B2):SET(B1+1,B2)
715 CP=1:GOSUB100
720 IFRC=2,S=S+1:CP=1:GOSUB100:GCTO720
730 IFRC=1,NEXTPT:GOTO800
740 HH=HH-10:IFHH<2ANDHH>-6,HH=2
750 IFD2=90RD2=440R(D2=34ANDD1<50),M2=-M2:GOSUB200:GOTO720ELSEIF
D1=220RD1=500RD1=78,M1=-M1:GOSUB200:GOTO720ELSEIFD1<50ANDD2<20,A
=M1:M1=-M2*2:M2=-A/2:GOSUB200:GOTO720ELSEA=M1:M1=M2*2:M2=A/2:GOS
UB200:GOTO720
800 GOSUB40:FORA=9TO44:SET(50,A):SET(51,A):NEXT:FORA=32TO44:SET(
78,A):SET(79,A):NEXT:FORA=52TO77:SET(A,44):NEXT:FORA=52TO111:SET
(A,9):NEXT:FORA=9TO23:SET(112,A):SET(113,A):NEXT:FORA=80TO95:SET
(A,32):NEXT:FORA=18TO23:SET(70,A):SET(71,A):SET(90,A-6)
810 SET(91,A-6):NEXT:FORA=24TO32:SET(116-A*2,A):SET(117-A*2,A):S
ET(158-A*2,A):SET(159-A*2,A):NEXT:HP=15838:PRINT@25,"HOLE # 6  P
AR 3";:P=3:FORPT=1TONP:S=1:PRINT@91,CHR$(30);:PRINT@91,Z(PT);"'S
 TURN";:CP=1:B1=RND(11)*2+52:B2=41:SET(B1,B2):SET(B1+1,B2)
815 GOSUB100
820 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO820
830 IFRC=1,NE)TPT:GOTO900
840 HH=HH-10:IFHH<2ANDHH>-6,HH=2
850 IFD2=90RD2=440R(D2=32ANDD1>79),M2=-M2:GOSUB200:GOTO820ELSEIF
D1=500RD1=780RD1=1120RD1=900R(D1=70ANDD2<23),M1=-M1:GOSUB200:GOT
O820ELSEA=M1:M1=-M2*2:M2=-A/2:GOSUB200:GOTO820
900 GOSUB40:FORA=30TO79:SET(A,44):NEXT:FORA=23TO43:SET(78,A):SET
(79,A):NEXT:FORA=30TO51:SET(A,30):SET(A,9):NEXT:FORA=20TO33:SET(
8,A):SET(9,A):SET(50,A+11):SET(51,A+11):NEXT:FORA=10TO29:SET(A,2
9+A/2):SET(A,24.5-A/2):NEXT:FORA=50TO79
910 SET(A,A/2-16):NEXT:FORA=50TO67:SET(A,A/2-9):NEXT:FORA=50TO59
:SET(A,A/2-1):NEXT:HP=16150:PRINT@25,"HOLE # 7  PAR 3";:P=3:FORP
T=1TONP:S=1:PRINT@91,CHR$(30);:PRINT@91,Z(PT);"'S TURN";:B1=RND(
11)*2+52:B2=41:SET(B1,B2):SET(B1+1,B2):CP=1:GOSUB100
920 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO920
```

```
930 IFRC=1,NEXTPT:GOTO1000
940 HH=HH-10:IFHH<2ANDHH>-6,HH=2
950 IFD1=78ORD1=80R(D1=50ANDD2)30),M1=-M1:GOSUB200:GOTO920ELSEIF
D2=30ORD2=9ORD2=44,M2=-M2:GOSUB200:GOTO920ELSEIFD1<40ANDD2<24,A=
M1:M1=-M2*2:M2=-A/2:GOSUB200:GOTO920ELSEA=M1:M1=M2*2:M2=A/2:GOSU
B200:GOTO920
1000 GOSUB40:FORA=50TO79:SET(A,44):SET(A,9):NEXT:FORA=33TO43:SET
(50,A):SET(51,A):SET(78,A):SET(79,A):NEXT:FORA=18TO24:SET(32,A):
SET(33,A):SET(96,A):SET(97,A):SET(56,A+1):SET(57,A+1):SET(72,A+1
):SET(73,A+1):NEXT:FORA=34TO49:SET(A,A/2+8)
1010 SET(A,34.5-A/2):SET(A+46,49.5-A/2):SET(A+46,A/2-7):NEXT:FOR
A=56TO64:SET(A,A/2-2):SET(A+9,58-A/2):NEXT:FORA=52TO57:SET(A,A/2
-10):SET(A+20,44.5-A/2):NEXT:HP=15904:PRINT@25,"HOLE # 8  PAR 3"
;:P=3:FORPT=1TONP:S=1:PRINT@91,CHR$(30);:B1=RND(11)*2+52
1015 PRINT@91,Z(PT);"'S TURN";:B2=41:SET(B1,B2):SET(B1+1,B2):CP=
1:GOSUB100
1020 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO1020
1030 IFRC=1,NEXTPT:GOTO1100
1040 HH=HH-10:IFHH<2ANDHH>-6,HH=2
1050 IFD2=44ORD2=9,M2=-M2:GOSUB200:GOTO1020ELSEIFD1=32ORD1=96ORD
1=72ORD1=56ORD1=50ORD1=78,M1=-M1:GOSUB200:GOTO1020ELSEIF(D1<78AN
DD1>63)OR(D1>70ANDD2>20)OR(D1<60ANDD2<20),A=M1:M1=-M2*2:M2=-A/2:
GOSUB200:GOTO1020ELSEA=M1:M1=M2*2:M2=A/2:GOSUB200
1060 GOTO1020
1100 GOSUB40:FORA=34TO95:SET(A,9):NEXT:FORA=50TO79:SET(A,44):NEX
T:FORA=33TO43:SET(50,A):SET(51,A):SET(78,A):SET(79,A):NEXT:FORA=
10TO33:SET(34,A):SET(35,A):SET(94,A):SET(95,A):NEXT:FORA=36TO49:
SET(A,33):SET(A+44,33):NEXT:FORA=48TO63
1110 SET(A,A/2-1):SET(A+18,54.5-A/2):NEXT:FORA=56TO61:SET(A,49.5
-A/2):SET(A+12,A/2-9):NEXT:HP=15904:PRINT@25,"HOLE # 9  PAR 4";:
P=4:FORPT=1TONP:PRINT@91,CHR$(30);:PRINT@91,Z(PT);"'S TURN";:S=1
:B1=RND(11)*2+52:B2=41:SET(B1,B2):SET(B1+1,B2):CP=1:GOSUB100
1120 IFRC=2,S=S+1:CP=1:GOSUB100:GOTO1120
1130 IFRC=1,NEXTPT:GOTO2000
1140 HH=HH-10:IFHH<2ANDHH>-6,HH=2
1150 IFD2=9ORD2=33ORD2=44,M2=-M2:GOSUB200:GOTO1120ELSEIFD1=34ORD
1=94OR(D1=78ANDD2>30)OR(D1=50ANDD2>30),M1=-M1:GOSUB200:GOTO1120E
LSEIF(D1>64ANDD2>22)OR(D1<64ANDD2<22),A=M1:M1=-M2*2:M2=-A/2:GOSU
B200:GOTO1120ELSEA=M1:M1=M2*2:M2=A/2:GOSUB200
1160 GOTO1120
```

End-game routine.

```
2000 CLS:PRINTCHR$(23);:PRINT@20,"FINAL SCORE:";:FORA=1TONP:PRINT
Z(A),S(A):NEXT:PRINT@900,"PLAY AGAIN (Y/N)";:INPUTZ:IFZ="Y",10
```

TRS-80® SWAT TABLE FOR:
MINIGOLF

LINES	SWAT CODE	LENGTH
10 - 60	YY	522
70 - 107	VS	523
108 - 205	KJ	447
210 - 300	MC	533
305 - 430	WR	503
435 - 520	IY	502
530 - 610	HW	573
615 - 700	WO	520
710 - 800	BW	649
810 - 900	WX	608
910 - 1000	UU	617
1010 - 1050	DJ	540
1060 - 1150	BK	658
1160 - 2000	ZL	104

NOTES:

SPACE RESCUE

by Matt Rutter
TRS-80 version by Alan J. Zett

***Space Rescue* is an arcade-style game for a 16K TRS-80.**

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 spacebar to launch your rocket. Then you must guide the rocket down to the landing pad by pressing the left and right arrows, being careful not to collide with any of the meteors. For finer control, press and hold CLEAR while pressing the arrows. To slow down, press the spacebar 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 spacebar will launch a missile that can destroy a meteor, for which you can receive points.

If one of your rockets collides with a meteor, 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 joystick, paddle, or keyboard 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       TRS-80 BASIC          SS
SS      'Space Rescue'         SS
SS    Author: Matt Rutter      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 variables.

```
10 CLS:GOTO1100
15 RANDOM:DEFINTA-X,Z:D1=0:SC=0:SL=3:U$=".#####."
20 CLS:S=51:D=-1:P1=6:P2=0:SHIP$=CHR$(158)+CHR$(191)+CHR$(173)
25 FORI=13TO15:PRINT@I*64+1,"X";:PRINT@I*64+62,"X";:NEXT
```

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

```
30 L=RND(39)+8:PRINT@894+L,CHR$(160)CHR$(184)CHR$(190)STRING$(3,
143)CHR$(189)CHR$(180)CHR$(144);:PRINT@958+L,STRING$(9,191);:B=0
:G=0
40 FORI=1TOD1*10+15:PRINT@(1+RND(9))*64+RND(63),CHR$(140);:NEXT:
PRINT@959+L,USINGU$;SC/1E5;:PRINT@0,;:FORAZ=1TOSL:SOUND37,100:SO
UND0,100:NEXT
```

Move mother ship back and forth.

```
50 S=S+D
60 PRINT@S-1," "CHR$(191)STRING$(4,179)CHR$(191)STRING$(4,179)CH
R$(191)" "CHR$(26)STRING$(12,24)" "CHR$(131)CHR$(191)CHR$(149)SH
IP$CHR$(170)CHR$(191)CHR$(131)" ";:IFS<20RS>50THEND=-D
65 IF(PEEK(14400)AND128)=128THEN80
70 FORW=1TO25:NEXTW:GOTO50
80 X=S+5:Y=1
```

Check for player input.

```
100 U=0:X1=X:P=PEEK(14400)AND96:IF(P=32ORP=64)ANDY>1.5GOSUB300
120 Y1=Y+1:IF(PEEK(14400)AND128)=128THENY1=Y+.5
```

Check for collision.

```
130 FORI=X1-1TOX1+1:A=PEEK(15360+I+(INT(Y1+.5)*64)):IFA=140THEN5
00
133 IFB=140THEN500
135 NEXTI
```

Move ship, and display flame if appropriate.

```
140 PRINT@X-1+(INT(Y+.5)*64),"   ";:PRINT@X+(INT(Y+1.5)*64)," ";
:X=X1:Y=Y1:PRINT@X-1+(INT(Y+.5)*64),SHIP$;
145 B=PEEK(15360+X+(INT(Y+1.5)*64)):IF(PEEK(14400)AND128)=128THE
NPRINT@X+(INT(Y+1.5)*64),"V";:PRINT@0,;:FORAZ=1TO5:SOUNDY*8,2:SO
UNDY*3,2:NEXT
```

Check for successful landing.

```
150 IFX=L+2ANDINT(Y+.5)=13THENPRINT@X+(INT(Y+1.5)*64),CHR$(143);
:SC=SC+50:PRINT@L+959,USINGU$;SC/1E5;:GOTO400
155 IFINT(Y)>12THEN500
160 GOTO100
```

Erase old meteors, and display new ones.

```
200 X=L+2:Y=13:FORI=2TO12:PRINT@I*64,STRING$(64,32);:NEXT
205 FORI=1TOD1*15+20:PRINT@(1+RND(9))*64+RND(63),CHR$(140);:NEXT
:PRINT@959+L,USINGU$;SC/1E5;
207 PRINT@894+L,"          ";
```

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

```
210 PRINT@0,;:FORAZ=1TO5:SOUNDY*8,2:SOUNDY*3,2:NEXT:X1=X:Y1=Y:P=
PEEK(14400)AND96:IFP=32ORP=64THENGOSUB300
220 C=C+1:IFC=3THENC=0:Y1=Y-1
```

Check for collision.

```
225 FORI=X1-1TOX1+1:A=PEEK(15360+I+(INT(Y1+.5)*64)):IFA=140THEN5
00
227 NEXT
```

Move rocket, and check for input.

```
230 PRINT@X-1+(INT(Y+.5)*64)," ";:PRINT@X+(INT(Y+1.5)*64)," ";
:X=X1:Y=Y1:PRINT@X-1+(INT(Y+.5)*64),SHIP$;:PRINT@X+(INT(Y+1.5)*6
4),"Y";
240 IF(PEEK(14400)AND128)=128ANDG=0ANDY>3.5THENGOSUB340
250 IFG=1THENGOSUB350
255 IFINT(Y)=1THEN270
260 GOTO210
```

Check for successful docking with mother ship.

```
270 IFX<>S+5THEN500
275 P2=P2+1
280 FORI=2TO15:PRINT@I*64+3,STRING$(57,32);:NEXT
```

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

```
285 IFP1=0THENP1=6:D1=D1+1:GOTO650
290 GOTO30
```

Subroutine to move ship according to input.

```
300 P=PEEK(14400)AND98:IFP=32ANDX>6THENX1=X-2
305 IFP=34ANDX>5THENX1=X-1
310 IFP=64ANDX<57THENX1=X+2
315 IFP=66ANDX<58THENX1=X+1
320 RETURN
```

Launch missile.

```
340 PRINT@0,;:FORAZ=10TO100STEP10:SOUNDAZ,3:NEXT:G=1:XM=X:YM=Y
```

Move missile and check for collision with meteors.

```
350 PRINT@XM+(INT(YM+.5)*64)," ";:YM=YM-1
360 A=PEEK(15360+XM+(INT(YM+.5)*64)):IFA=140THENPRINT@XM+(INT(YM
+.5)*64)," ";:G=0:SC=SC+20:PRINT@959+L,USINGU$;SC/1E5;:RETURN
370 PRINT@XM+(INT(YM+.5)*64),"!";:IFYM<3THENPRINT@XM+(INT(YM+.5)
*64)," ";:G=0
380 RETURN
```

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

```
400 Y=19-P1:X=62:X1=58:X2=L+8:IFP1<4THENY=16-P1:X=1:X1=2:X2=L-4
```

```
410 FORI=1TO4:FORJ=43TO88STEP45:PRINT@Y%64+X,CHR$(J);:FORW=1TO30
:NEXTW,J
420 PRINT@Y%64+X1,"YAY!";:IFI/2=INT(I/2)THENPRINT@Y%64+X1,"    "
;:PRINT@0,;:FORAZ=1TO2:FORAY=77TO7STEP-7:SOUNDAY,4:NEXT:FORAY=7T
077STEP7:SOUNDAY,4:NEXT:NEXT
430 NEXTI:PRINT@Y%64+X," ";
440 FORI=X+SGN(X1-X) TO X2 STEP SGN(X2-X)
450 FORJ=43TO88STEP45:PRINT@960+I,CHR$(J);:FORW=1TO20:NEXTW
455 PRINT@0,;:SOUND100,0:SOUND200,0:NEXTJ:PRINT@960+I," ";:NEXTI
460 X=I+SGN(X1-X2):Y=15:FORI=1TO2:X=X+SGN(L-X1):Y=Y-1
465 FORJ=43TO88STEP45:PRINT@Y%64+X,CHR$(J);:FORW=1TO20:NEXTW:PRI
NT@Y%64+X," ";:NEXTI
470 P1=P1-1:U=1:GOTO200
```

Explosion routine.

```
500 REM
520 X1=X-2:X2=X+2:X0=X:YS=2:PRINT@X-1+(INT(Y+.5)%64),"   ";:PRIN
T@X+(INT(Y+1.5)%64)," ";
530 FORI=YTO14:PRINT@X2+I%64,CHR$(170)CHR$(173);:PRINT@X1+I%64,C
HR$(158)CHR$(149);:IFU=1THENPRINT@X0+I%64,"X";
540 PRINT@0,;:FORAZ=1TO10:SOUNDRND(255),RND(2)-1:NEXT:YS=YS-.25:
IFYS<0THENYS=0
550 PRINT@X1+I%64,"  ";:PRINT@X2+I%64,"  ";:IFU=1THENPRINT@X0+I%
64," ";
560 X1=X1-YS:IFX1<5THENX1=58
570 X2=X2+YS:IFX2>58THENX2=5
580 NEXTI:SL=SL-1:IFSL=0THEN600
590 GOTO280
```

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

```
600 CLS:PRINT@409,"% GAME OVER %";:PRINT@533,"YOUR SCORE IS ";US
INGU$;SC/1E5
610 PRINT@658,"PRESS <ENTER> TO PLAY AGAIN";
620 IFINKEY$<>CHR$(13)THEN620ELSE30TO15
```

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

```
650 CLS
660 PRINT@411,"% BONUS %":PRINT@536,"SCORE = ";USINGU$;SC/1E5
670 FORI=1TOP2:PRINT@664+I%2,"X";:SC=SC+50:PRINT@544,USINGU$;SC/
1E5
680 PRINT@0,;:FORAZ=1TO10:SOUNDI%15+30,3:SOUNDI%15+60,3:NEXT:FOR
W=1TO50:NEXTW:NEXTI
```

```
690 PRINT@0,;:FORAZ=1TO100:SOUND100-AZ,2:SOUNDAZ,2:NEXT:IFP2=6TH
ENPRINT@406,"*** BONUS SHIP! ***";:SL=SL+1:FORP2=1TO4:FORAZ=80TO
185STEP7:SOUNDAZ,1:SOUNDAZ+50,1:NEXT:NEXT
```

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

```
695 FORW=111TO111STEP3:FORAZ=WTOW-7STEP-1:SOUNDAZ,2:NEXT:NEXT
700 GOTO20
```

Initialize the sound routine.

```
1000 Z=0:FORX=1TO158:READ Y:Z=Z+Y:NEXT:IFZ<>15204THENCLS:PRINT"D
ATABASE ERROR IN LINES 1030-1080; CHECK LISTING.":PRINT:LIST 103
0-1080ELSEY=86:X=255:POKE-1,0:IFPEEK(-1)<>0THENX=191:POKE-16385,
0:IFPEEK(-16385)<>0THENX=127
1005 POKE 16562,X:POKE 16561,Y:CLEAR500:A1=PEEK(16561)+2:A2=PEEK
(16562):A=A1+A2*256:Z=A-1:FORX=1TO158:Z=Z+1:Z=Z+65536*(Z>32767)
1010 READY:IFY<0THENY=A1+ABS(Y):POKEZ,Y+256*(Y>255):Z=Z+1:POKEZ,
A2-(Y>255):NEXTELSEPOKEZ,Y:NEXT
1015 IFPEEK(16396)=201POKE16526,A1:POKE16527,A2ELSECMD"T":DEFUSR
=A1+(A2+256*(A2>127))*256:POKE14308,0
1020 IFPEEK(16807)+PEEK(16808)*256<>A+24THENA=USR(0)
1025 GOTO15
1030 DATA58,166,65,50,-164,42,167,65,34,-165,62,195,50
1035 DATA166,65,33,-24,34,167,65,201,245,123,254,2,40,4,254
1040 DATA16,32,79,229,213,42,230,64,126,183,32,4,35,35,35,35
1045 DATA215,6,5,17,-156,26,190,32,104,19,35,16,248,43,215
1050 DATA43,34,230,64,241,241,241,241,197,213,215,205,55,35
1055 DATA229,205,127,10,42,33,65,34,-167,225,215,43,34,230,64
1060 DATA35,205,55,35,43,229,205,127,10,42,33,65,58,-167,60
1065 DATA183,87,24,4,24,48,24,44,66,62,1,211,255,16,252,66,62
1070 DATA2,211,255,16,252,58,64,56,230,4,32,7,124,181,40,3,43
1075 DATA24,228,175,50,154,64,225,209,193,215,195,30,29,83,79
1080 DATA85,78,68,209,225,241
```

Draw the opening display on the screen.

```
1100 CLS:PRINTCHR$(23);
1110 PRINT@208,CHR$(191)STRING$(13,143)CHR$(191):PRINT@272,CHR$(
191)"  S P A C E  "CHR$(191):PRINT@336,CHR$(191)" R E S C U E "C
HR$(191):PRINT@400,CHR$(191)STRING$(13,188)CHR$(191)
1120 PRINT@648,"ORIGINAL BY MATT RUTTER":PRINT@716,"S-80 BY ALAN
 J ZETT"
1130 GOTO1000
```

TRS-80® SWAT TABLE FOR:
SPACE RESCUE

LINES	SWAT CODE	LENGTH
10 - 65	MZ	516
70 - 160	ZJ	508
200 - 260	WI	494
270 - 350	NE	289
360 - 455	FB	537
460 - 590	ME	491
600 - 1000	YB	661
1005 - 1045	YR	524
1050 - 1110	UJ	512
1120 - 1130	QC	74

NOTES:

Maze Sweep

by David Bohlke **TRS-80 version by James Garon**

***Maze Sweep* is an arcade-style game for a TRS-80 with 16K RAM.**

In *Maze Sweep,* your man, represented by an asterisk (*), must try to sweep up the targets scattered through a maze. The computer first creates the maze, then places 25 targets in it. The targets appear as O's.

Using the four arrow keys, try to hit as many targets as you can before the timer runs out. Each target is worth 40 points, giving a possible maximum of 1000 points. If you are skillful enough to hit all 25 targets, you will earn a bonus of ten times the number of ticks left on the timer. Note that time passes at a fairly reasonable rate as long as you keep moving. If you stop to catch your breath, however, the timer will count down like crazy.

To hear the sound effects that accompany this game, connect the cable that normally plugs into the "auxiliary" jack of the cassette recorder to an amplifier. Alternatively, you can use a cassette recorder as an amplifier if you have an external speaker. Just leave the "auxiliary" cable connected to the recorder, plug the speaker into the recorder's "earphone" jack, and start it running in "record" mode.

Programming Notes

The first unusual thing an alert reader will notice is in line 20. Yes, simple (non-array) variables can be placed in a DIM statement without causing an error. All such variables are given memory space at DIM time, rather than at the time they are first used on the left side of an equals sign. Line 20 DIMs every simple variable in the program. Once an array variable has been DIMensioned, whenever a new simple variable is mentioned in the program, something unpleasant happens: the array is moved to make room for the new variable. If the array is large, this relocation can waste considerable time.

In this case, time is not a concern. The important issue here is that M(13) must not move *at all.* Why? Because this particular array contains the Machine Language sound-effects subroutine. When a USR call jumps to where the routine was, and doesn't find it there, the results can be catastrophic.

Line 30 contains my high score. Beat it if you can.

Lines 40-60 set up the sound-effects subroutine. This is stored in an integer array, as mentioned. If you use this idea yourself, it is essential that the array be an *integer* array. Either use a DEFINT statement or append a % symbol to the array's name. Items in an integer array each take up two bytes of memory; thus, two bytes of Machine Language code can reside in each element of the array. To determine what single number to put in the first element of the array, take the first two decimal numbers of your Machine Language routine. Let's say these are 205 and 127. Multiply the second number by 256, and add the result to the first number. In our example, this yields 32717. If the result is less than 32768, use it as-is. If it is equal to or greater than 32768, subtract 65536 from it.

Continue by this method, combining the third and fourth decimal numbers,

the fifth and sixth, and so on, until you finish the last pair. If your routine contains an odd number of bytes, pretend that there is a zero after the last byte. Read these numbers into an integer array, get the VARPTR of the zero element of the array, and that's the entry point for your USR routine. (See lines 50-60.) One advantage of this method is that it does not create strings of garbage in the middle of your program (as does POKEing into a string).

The construction of the maze begins in line 200. From a starting point at the center of the screen, one of four directions is chosen at random. If the way is clear, a line is drawn in that direction. If not, a new direction is chosen. If all directions are blocked, the computer backs up to the most recent unblocked position and continues drawing. When this backing-up procedure causes the computer to return to the starting point, the maze is complete.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS       TRS-80 BASIC          SS
SS       'Maze Sweep'          SS
SS    Author: David Bohlke     SS
SS  Translator: James Garon    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 CLEAR 500:DEFINT A-Z:V=15360:N=191:M$=CHR$(N)
20 DIM B,B1,B2,I,J,K,M,P,Q,SC,SC$,T,Z,M(13),P(200)
```

The author's high score. Beat it if you can!

```
30 SC(10)=6640:SC$(10)="JAMES GARON"
```

Set up sound routine.

```
40 DATA 32717,19722,15940,26881,-45,8237,15613,-11415,11775,-736
,4160,-20498,-45,201
50 FOR I=0 TO 13:READM(I):NEXT:I=VARPTR(M(0)):J=I/256:K=I-J*256
60 IF PEEK(16396)=201 POKE 16526,K:POKE 16527,J ELSE CMD"T":DEFU
SR=I:POKE 14309,0
```

Draw the border of the maze and display the time, title, and score.

```
70 CLS:PRINT" TIME: 1000"TAB(24)"MAZE SEARCH","SCORE: 0"
80 PRINT" "STRING$(63,N);:FOR I=1 TO 13
90 PRINT" "CHR$(N)CHR$(253)CHR$(N);:NEXT
```

```
100 PRINT" "STRING$(62,N);:POKE 16383,N
```

Construct the maze.

```
200 P=64*7+31:PRINT@P,M$;:P(0)=P:K=1
210 IF PEEK(V+P+2)-N THEN 260
220 IF PEEK(V+P-2)-N THEN 260
230 IF PEEK(V+P+128)-N THEN 260
240 IF PEEK(V+P-128)-N THEN 260
250 GOTO 350
260 ON RND(4) GOTO 270,280,290,300
270 Q=-1:GOTO 310
280 Q=64:GOTO 310
290 Q=1:GOTO 310
300 Q=-64
310 IF PEEK(V+P+Q*2)=N THEN 260
320 PRINT@P+Q,M$;:PRINT@P+Q*2,M$;:I=USR(1281+K)
330 P=P+Q*2:P(K)=P:K=K+1
340 GOTO 210
350 K=K-1:P=P(K):PRINT@P," ";:I=USR(1281+K):PRINT@P,CHR$(N);
360 IF K=0 THEN 400
370 GOTO 210
```

Position the 25 targets (O) and the player (*), after disabling the BREAK key — in any game using the screen as memory, accidentally pressing BREAK is disastrous.

```
400 GOSUB 9000:FOR I=1 TO 26
410 P=130+RND(11)*64+RND(59):IF PEEK(V+P)-32 THEN 410 ELSE PRINT
@P,"O";:NEXT
```

Accept arrow-key input from the keyboard and allow the player to move. Decrement the time, and test for hitting a target, running out of time, or hitting the last target.

```
600 T=1000:SC=0:FOR I=1 TO 99:PRINT@P,"*";:PRINT@USR(P),CHR$(143
);:NEXT
610 PRINT@P,"*";:I=USR(513+T/4)
620 T=T-1:PRINT@6,T;:IF T=0 THEN 800
630 POKE 16444,0:I$=INKEY$:IF I$="" THEN 620 ELSE Z=ASC(I$):Q=P
640 IF Z=8 THEN Q=P-1
650 IF Z=9 THEN Q=P+1
660 IF Z=10 THEN Q=P+64
670 IF Z=91 THEN Q=P-64
680 M=PEEK(Q+V):IF M=N THEN 610
```

```
690 IF M=32 OR M=42 ELSE SC=SC+40:PRINT@55,SC;:PRINT@P," ";:PRIN
T@Q,"*";:GOSUB 2500
700 PRINT@P," ";:IF SC<1000 THEN P=Q:GOTO 610
```

Game is over. Award any bonus that the player has won.

```
800 PRINT@Q,"*";:T=T*10:SC=SC+T:PRINT@1,"BONUS";:GOSUB 2520
810 FOR I=1 TO 35:PRINT@I," BONUS:"T;:FOR J=1 TO 30:NEXT:NEXT
820 PRINT@I,CHR$(203);:PRINT@55,SC;:GOSUB 2510
```

Maintain and update the top ten players and their scores.

```
2000 M=0:FOR I=1 TO 10
2010 IF SC>SC(I) THEN M=I
2020 NEXT:IF M=0 THEN 2100
2030 PRINT@153,"CONGRATULATIONS;";
2040 PRINT@208,"YOUR SCORE IS ONE OF THE TOP TEN.";
2050 PRINT@278,"PLEASE ENTER YOUR NAME";
2060 SC$="":PRINT@479,;:INPUT SC$:IF SC$=""THEN 2060
2070 CLS:SC$=LEFT$(SC$,15)
2080 FOR I=1 TO M:SC(I-1)=SC(I):SC$(I-1)=SC$(I):NEXT
2090 SC(M)=SC:SC$(M)=SC$
2100 CLS:PRINT CHR$(23)"TOP TEN SCORES":PRINT
2110 FOR I=10 TO 1 STEP -1
2120 IF SC$(I)>"" THEN PRINT SC$(I),SC(I) ELSE PRINT"...","
.
2130 NEXT
2140 GOSUB 9010:PRINT:PRINT"PRESS =ENTER= TO PLAY AGAIN"
2150 IF INKEY$<>CHR$(13) THEN 2150 ELSE 70
```

Special sound effects.

```
2500 FOR K=0 TO 2:J=USR(4778) OR USR(4742):NEXT:RETURN
2510 FOR K=0 TO 4:J=USR(7764) OR USR(7746):NEXT:RETURN
2520 J=USR(8814):FOR K=0 TO 1:J=USR(8792) OR USR(8778) OR USR(87
65) OR USR(8778) OR USR(10328) OR USR(10350):NEXT:RETURN
```

Disable the BREAK key.

```
9000 B=16396:B1=PEEK(B):B2=PEEK(B+1):POKE B,60:POKE B+1,201:RETU
RN
```

Re-enable the BREAK key.

```
9010 POKE B,B1:POKE B+1,B2:RETURN
```

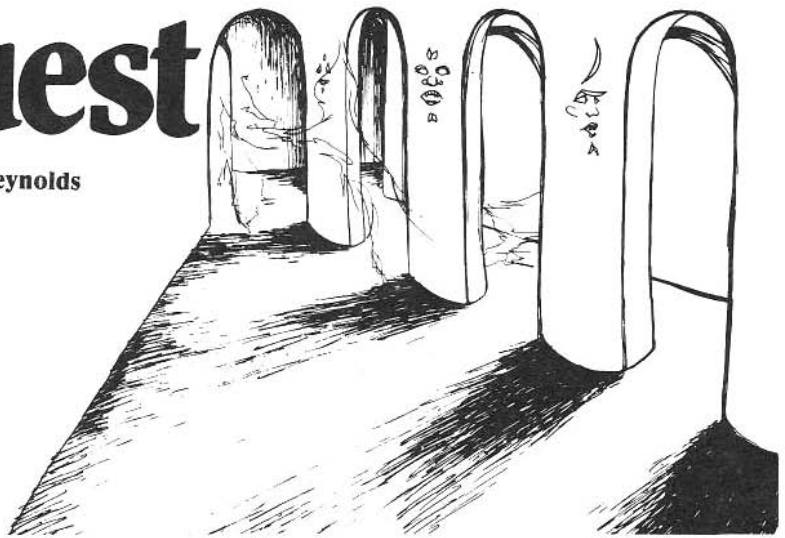
TRS-80® SWAT TABLE FOR:
MAZE SWEEP

LINES	SWAT CODE	LENGTH
10 - 210	GS	508
220 - 330	HG	254
340 - 650	BU	355
660 - 2030	VH	379
2040 - 2150	IE	394
2500 - 9010	DK	237

NOTES:

Quest

by Brian Reynolds



***Quest 1* is a graphic adventure game for a TRS-80 with 16K 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. If you have a Model III, your character will look different, depending on his/her gender. On the Model I, all characters appear as "@" signs.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:

If you have a numeric keypad, the numbers are convenient.

8 or up arrow: Move up.
4 or left arrow: Move left.
6 or right arrow: Move right.
2 or down arrow: 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%.)

Below is a complete inventory of the monsters, with their wound ratings. These ratings represent the monster's strength, relative to your initial strength. If you are attacked, by a skeleton for example, it can inflict wounds of up to 20% on you with each hit. And, it takes more to kill a monster with a high rating than one with a low rating.

Skeleton: 20%
Orc: 30%
Zombie: 40%
Ghoul: 50%
Spider: 70%
Mummy: 80%
Giant: 90%
Wraith: 99%

Note: Following the listing of *Quest 1* is a listing entitled *Quest 2 Database.* This consists of statements that provide you with a new dungeon, for when you get tired of the original one.

To use the listing, type it in as a separate program. The *SWAT* table for *Quest 2 Database* is for it alone. To save it on disk, type SAVE "QUEST2/DAT",A. The "A" option allows you to merge the program in with the main program. To do that, load *Quest 1,* then type MERGE "QUEST2/DAT". That's all there is to it. If you have cassette only, simply load the original program, and type the new lines over it.

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.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS       TRS-80 BASIC          SS
SS        'Quest  1'           SS
SS  Author:  Brian Reynolds    SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

Print title page.

```
1 CLS:CLEAR400:IFPEEK(664)=58ANDPEEK(665)=16THENTRS80MODEL=3ELSE
TRS80MODEL=1
2 PRINT@476,"QUEST 1";:IFTRS80MODEL=3THENPOKE16420,1
3 A$="                * * * * * * * * * *  Q U E S T  1  W
 A S  W R I T T E N  B Y  B R I A N  R E Y N O L D S  * * *
 * * * * * *  (HAVE FUN!)  "
4 FORX=1TOLEN(A$)-14:PRINT@537,MID$(A$,X,14);:FORY=1TO30:NEXT:NE
XT:PRINT
5 ON ERROR GOTO 30000
6 FORX=1TO1000:NEXT
7 PRINT
8 IFTRS80MODEL=1THENYY$="@"ELSEIFTRS80MODEL=3THENYY$=CHR$(253)
```

Data for monsters and treasures.

```
100 DATA "WORTHLESS ODDS & ENDS",0,0,"A BAG FULL OF COPPER COINS
",1,3,"A SMALL BRASS STATUETTE",2,5,"A BAG FULL OF VARIOUS COINS
",3,7,"A PURSE FULL OF GOLD COINS",5,12,"3 GOLD NUGGETS     ",8,
17,"4 SMALL TURQUOISES",7,15,"LARGE RUBY        ",15,30
105 DATA "A *HUGE* SAPPHIRE",150,150,"A HEALING POTION",10,0,"A
QUIVER OF 10 MAGIC ARROWS",15,0,"A QUIVER OF 10 NORMAL ARROWS",1
0,0
110 DATA"SKELETON","S",2,"ORC","O",3,"ZOMBIE","Z",4,'GHOUL","G",
6,"HUGE SPIDER","H",7,"MUMMY","M",8,"GIANT","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
```

Initialize the variables.

```
600 DIM MN$(8),M$(8),MS(8),R1(58),R2(58,4),M1(58),M2(58),T1(58),
T$(12),EP(12),GP(12)
603 FORX=1TO12:READT$(X),EP(X),GP(X):NEXTX
605 FORX=1TO8:READMN$(X),M$(X),MS(X):NEXT
610 FORX=1TO58:READR1(X):FORY=1TO4:READR2(X,Y):NEXTY
615 READM1(X),M2(X),T1(X):NEXTX
620 PM=1:A1=1000:A2=1000:W=1:PT=2
625 IFTRS80MODEL=3THENPOKE16409,1
```

Use an old character?

```
800 IFB1=1THENGOSUB20000
805 IFB1=1THENB1=0:GOTO900
810 INPUT"DO YOU WISH TO USE AN OLD CHARACTER";A$:IFLEFT$(A$,1)<
>"Y"THENGOSUB21000:GOTO900
811 IFTRS80MODEL=3THENPOKE16409,0
```

```
812 INPUT"NAME ";NM$
813 IFTRS80MODEL=3THENPOKE16409,1
815 INPUT"STRENGTH ";ST:IFST>200RST<3THEN815
820 INPUT"DEXTERITY ";DX:IFDX>200RDX<3THEN820
825 INPUT"WOUNDS ";W:W=W/100:IFW<.1ORW>1THEN825
830 INPUT"EXPERIENCE ";EP:INPUT"GOLD: ";GP
835 INPUT"IS (S)HE AN ELF";A$:IFLEFT$(A$,1)="Y"THENRC=1
836 IFRC=0THENINPUT"IS (S)HE A DWARF";A$:IFLEFT$(A$,1)="Y"THENRC
=2
840 INPUT"MAGIC ARROWS ";A2:INPUT"NORMAL ARROWS ";AI
845 INPUT"HEALING POTIONS ";PT
846 INPUT"HOLY WATER ";HW
847 IFTRS80MODEL=3THENINPUT"IS THIS CHARACTER FEMALE";A$:IFLEFT$
(A$,1)="Y"THENYY$=CHR$(254)
```

Ask whether to load an old game.

```
850 INPUT"DO YOU WISH TO LOAD IN AN OLD GAME";A$:IFLEFT$(A$,1)<>
"Y"THEN900
860 INPUT"FROM CASSETTE OR DISK ";A$:IF LEFT$(A$,1)="C"THENINPUT
"PRESS ENTER TO BEGIN LOAD ";A$:GOTO880
862 IF LEFT$(A$,1)<>"D"THEN860
870 OPEN"I",1,"QUEST/DAT"
872 FORX=1TO58:INPUT#1,M1(X),M2(X),T1(X):NEXT
876 CLOSE:GOTO900
880 FORX=1TO58:INPUT#-1,M1(X),M2(X),T1(X):NEXT
```

Marketplace and bargaining routine.

```
900 CLS:PRINT"GOLD: ";GP
901 PRINT"YOU ARE AT THE MARKET. PRICES HERE ARE:"
902 PRINT
903 PRINT"1. MAGIC ARROW ------------------------------------
-- 2 GOLD"
904 PRINT"2. 4 NORMAL ARROWS --------------------------------
-- 1 GOLD"
905 PRINT"3. HEALING POTION ---------------------------------
- 15 GOLD"
906 PRINT"4. HOLY WATER --------------------------------------
-- 3 GOLD"
910 PRINT" OK, ";NM$;", WHAT ITEM WOULD YOU LIKE (NUMBER)";:INPU
TIT:IFIT>4ORIT<0PRINT"I DON'T SELL THAT.":GOTO910
911 IFIT=0THEN990
912 IFIT=1THENP1=2ELSEIFIT=2THENP1=1ELSEIFIT=3THENP1=15ELSEIFIT=
4THENP1=3
```

```
915 PRINT" AT ";P1;" GOLD APIECE, HOW MANY WILL YOU BUY";:INPUTN
M:IFNM<1THENPRINT"VERY FUNNY. I DO NOT BUY THINGS, I SELL THEM."
:GOTO915
920 P1=P1*NM
921 OP=P1
925 PRINT"THE PRICE NOW COMES TO ";P1;" GOLD."
930 PRINT"HOW MUCH WILL YOU GIVE ME, ";NM$;:INPUTA
935 IFA<(OP/10)THENPRINT"FORGET IT!!!!":GOTO901
940 IFA<OP/2THENPRINT"NOT INTERESTED.":GOTO901
941 IFA>=P1THENPRINT"YOU GOT A DEAL!!!":GOTO950
942 Y=A/P1:X=RND(0):IFX>YTHENPRINT"NOT INTERESTED":P1=INT((OP+P1
)/2):GOTO930
945 P1=INT((P1*2+A)/3):PRINT"HOW ABOUT ";P1;", ";NM$;"?":GOTO930
950 IFGP<P1THENPRINT"WHAT!!! CAN'T PAY YER DEBTS??? YOU'LL BE TH
ROWN IN PRISON FOR  THIS, ";NM$;"!!!":END
955 GP=GP-P1:PRINT"YOU NOW HAVE ";GP;" GOLD, ";NM$;"."
960 IFIT=1THENA2=A2+NM
965 IFIT=2THENA1=A1+NM*4
970 IFIT=3THENPT=PT+NM
975 IFIT=4THENHW=HW+NM
980 GOTO901
```

Enter dungeon; check for too many arrows.

```
990 PRINT"OK, ";NM$;", PRESS <ENTER> TO GO INTO THE DUNGEON."
991 EL=0
992 IFEP>100THENEP=EP-100:EL=EL+100:FORX=1TO58:M2(X)=M2(X)*1.1:N
EXT:GOTO992
993 EP=EP+EL
994 IFEL>500THENFORU=ELTO500STEP-100:FORX=1TO58:M2(X)=M2(X)/1.1:
NEXT:NEXT
995 INPUTA$:CLS
997 A3=0:A4=0
998 IFA2>ST*2THENA4=A2-ST*2:A2=ST*2:PRINT"MORE THAN ";ST*2;" MAG
IC ARROWS WOULD WEIGH YOU DOWN.":FORX=1TO1000:NEXT
999 IFA1>ST*2THENA3=A1-ST*2:A1=ST*2:PRINT"MORE THAN ";ST*2;" ARR
OWS WOULD WEIGH YOU DOWN.":FORX=1TO1000:NEXT
```

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

```
1000 CLS
1001 IFRM=0THENRM=1ELSEIFRM=1THENINPUT"DO YOU WISH TO LEAVE THE
DUNGEON";A$:IFLEFT$(A$,1)="Y"THEN800ELSECLS
1005 ONRM(RM)GOSUB10000,11000
```

```
1010 IFT1(RM)>0THENTX=RND(39-23)+23:TY=RND(4)+5:PRINT@TX+64*TY,"
*";
1015 X5=31:Y5=8
1020 IFI$="8"THENY5=14ELSEIFI$="2"THENY5=1ELSEIFI$="6"THENX5=1EL
SEIFI$="4"THENX5=62
1025 PRINT@X5+64*Y5,YY$;
1030 IFM2(RM)>=1THENWX=RND(39-23)+23:WY=RND(4)+5
1031 MS=MS(M1(RM))/10
1050 IFM2(RM)>=1THENPRINT@WX+64*WY,M$(M1(RM));
```

Print player status. Check for wandering monsters.

```
1055 PRINT@0,"ARROWS: ";A1;:PRINT@192,"M. ARROWS: ";A2;
1060 PRINT@960,"ST=";ST;" DX=";DX;:PRINT@64,"WOUNDS: ";LEFT$(STR
$(W*100),6);"%   ";
1061 PRINT@128,"ROOM: ";RM;
1062 PRINT@832,"HEALING POTIONS: ";PT;
1063 PRINT@896,"HOLY WATER: ";HW;
1065 IFM2(RM)>=1THENPRINT@42,"MONSTER: ";MN$(M1(RM));
1070 IFM2(RM)<1THENPRINT@42,STRING$(64-42,128);
1075 IFM2(RM)>1THENPRINT@42+64,"NUMBER: ";INT(M2(RM));:ELSEPRINT
@42+64,STRING$(20,128);
1080 PRINT@704,"EX POINTS: ";INT(EP);
1085 PRINT@768,"GOLD: ";GP;
1086 IFM2(RM)=0ANDRND(100)=1THENFORX=1TO10:PRINT@42,"WANDERING M
ONSTER!";:FORY=1TO50:NEXT:PRINT@42,"                    ";:FORY=
1TO50:NEXT:NEXT:M2(RM)=RND(3):M1(RM)=RND(8):GOTO1030
```

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

```
1090 FORX=1TO(DX*10)-EP:A$=INKEY$:IFA$=""THENNEXTELSEX=3550:NEXT
1093 IFT1(RM)>0THENPRINT@TX+64*TY,"*";
1095 IFA$<>""THENIFA$="["THENI$="8"ELSEIFA$=CHR$(10)THENI$="2"EL
SEIFA$=CHR$(9)THENI$="6"ELSEIFA$=CHR$(8)THENI$="4"ELSEI$=A$
1100 IFI$="8"THENGOSUB15100
1105 IFI$="2"THENGOSUB15200
1110 IFI$="6"THENGOSUB15300
1115 IFI$="4"THENGOSUB15400
1120 IFI$="H"THENI$="":IFPT>0THENPT=PT-1:W=1
1125 IFI$="M"ANDA2>0THENI$="":A2=A2-1:GOSUB15500
1130 IFI$="N"ANDA1>0THENI$="":A1=A1-1:GOSUB15600
1135 IFI$="F"THENGOSUB16000
1140 IFI$="O"THENGOSUB17000
1145 IFI$="I"ANDHW>0THENI$="":HW=HW-1:GOSUB 18000
```

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

```
1200 IFM2(RM)<1THEN1030
1201 IFMS<=0THENFORX=191TO128STEP-1:POKE15360+WX+(64*WY),X:NEXT:
M2(RM)=M2(RM)-1:EP=EP+M5(M1(RM)):GOTO 1030
1205 IFWX>X5THENMX=-1ELSEIFWX<X5THENMX=1ELSEMX=0
1210 IFWY>Y5THENMY=-1ELSEIFWY<Y5THENMY=1ELSEMY=0
1215 PRINT@WX+64*WY," ";
1220 IFPEEK(15360+WX+MX+64*WY)=128ORPEEK(15360+WX+MX+64*WY)=32TH
ENWX=WX+MX
1225 IFPEEK(15360+64*(WY+MY)+WX)=128ORPEEK(15360+64*(WY+MY)+WX)=
32THENWY=WY+MY
1230 IF(ABS(WX-X5)>1)OR(ABS(WY-Y5)>1)THEN1050
1235 X=RND(0):IFX>MSTHEN1050
1240 X=RND(0)*MS
1245 W=W-X:IFW<0THEN5000
1250 GOTO 1050
```

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

```
5000 FORX=191TO128STEP-1:POKE15360+X5+64*Y5,X:NEXT:FORX=1TO1000:
NEXT:CLS
5005 PRINT"WELCOME TO HEAVEN, ";NM$;"!!!"
5010 PRINT"I HOPE YOU ENJOYED YOUR SHORT LIFETIME IN WHICH"
5015 PRINT"YOU ACCUMULATED ";GP;" GOLD AND ";EP;" EXPERIENCE POI
NTS."
5020 PRINT:PRINT:INPUT"WOULD YOU LIKE TO BE REINCARNATED AS A NE
W CHARACTER";A$:IFLEFT$(A$,1)="N"THENENDELSERUN
```

Subroutine to draw a passage/intersection.

```
10000 REM DRAW HALLWAY (TYPE #1)
10005 X1=R2(RM,1)
10010 IFX1>0THENFORX=0TO320STEP64:PRINT@X+23,CHR$(191);:PRINT@X+
40,CHR$(191);:NEXTELSEPRINT@343,STRING$(18,188);
10015 X1=R2(RM,2)
10020 IFX1>0THENFORX=640TO960STEP64:PRINT@X+23,CHR$(191)+STRING$
(16,128)+CHR$(191);:NEXT:ELSEPRINT@663,STRING$(18,143);
10025 X1=R2(RM,3)
10030 IFX1>0THENFORX=41TO63:PRINT@320+X,CHR$(188);:PRINT@640+X,C
HR$(143);:NEXT:ELSEFORX=384TO576STEP64:PRINT@X+40,CHR$(191);:NEX
T
10035 X1=R2(RM,4)
10040 IFX1>0THENFORX=0TO22:PRINT@320+X,CHR$(188);:PRINT@640+X,CH
R$(143);:NEXT:ELSEFORX=384TO576STEP64:PRINT@X+23,CHR$(191);:NEXT
10045 RETURN
```

Subroutine to draw a chamber/room.

```
11000 DRAW CHAMBER (TYPE #2)
11005 PRINT@192+16,CHR$(191)+STRING$(7,143)+STRING$(16,128)+STRI
NG$(7,143)+CHR$(191);
11010 PRINT@768+16,CHR$(191)+STRING$(7,188)+STRING$(16,128)+STRI
NG$(7,188)+CHR$(191);:FORX=192TO320STEP64:PRINT@X+16,CHR$(191);:
PRINT@X+47,CHR$(191);:NEXT:FORX=640TO768STEP64:PRINT@X+16,CHR$(1
91);:PRINT@X+47,CHR$(191);:NEXT
11015 X1=R2(RM,1)
11020 IFX1>0THENFORX=0TO128STEP64:PRINT@X+23,CHR$(191)+STRING$(1
6,128)+CHR$(191);:NEXT:ELSEPRINT@192+24,STRING$(16,143);
11025 X1=R2(RM,2)
11030 IFX1>0THENFORX=832TO960STEP64:PRINT@X+23,CHR$(191)+STRING$
(16,128)+CHR$(191);:NEXT:ELSEPRINT@768+24,STRING$(16,188);
11035 X1=R2(RM,3)
11040 IFX1>0PRINT@320+48,STRING$(16,188);:PRINT@640+48,STRING$(1
6,143);:ELSEFORX=384TO576STEP64:PRINT@X+47,CHR$(191);:NEXT
11045 X1=R2(RM,4)
11050 IFX1>0PRINT@320,STRING$(16,188);:PRINT@640,STRING$(16,143)
;:ELSEFORX=384TO576STEP64:PRINT@X+16,CHR$(191);:NEXT
11055 RETURN
```

Subroutines to move player about on the screen.

```
15100 IFY5=0THEN15105ELSEM=PEEK(15360+X5+((Y5-1)*64)):IFM=32ORM=
128THEN15105ELSERETURN
15105 PRINT@X5+64*Y5," ";
15110 Y5=Y5-1:IFY5<1THENRM=R2(RM,1):GOTO1000
15120 PRINT@X5+64*Y5,YY$;:RETURN
15200 IFY5=15THEN15205ELSEM=PEEK(15360+X5+((Y5+1)*64)):IFM<>32AN
DM<>128THENRETURN
15205 PRINT@X5+64*Y5," ";
15210 Y5=Y5+1:IFY5>14THENRM=R2(RM,2):GOTO1000
15220 PRINT@X5+64*Y5,YY$;:RETURN
15300 IFX5>61THEN15305ELSEM=PEEK(15360+X5+1+64*Y5):IFM<>32ANDM<>
128THENRETURN
15301 M=PEEK(15360+X5+2+64*Y5):IFM<>32ANDM<>128THENRETURN
15305 PRINT@X5+64*Y5," ";
15310 X5=X5+2:IFX5>61THENRM=R2(RM,3):GOTO1000
15320 PRINT@X5+64*Y5,YY$;:RETURN
15400 IFX5<2THEN15405ELSEM=PEEK(15360+X5-1+64*Y5):IFM<>32ANDM<>1
28THENRETURN
15401 M=PEEK(15360+X5-2+64*Y5):IFM<>32ANDM<>128THENRETURN
15405 PRINT@X5+64*Y5," ";
15410 X5=X5-2:IFX5<2THENRM=R2(RM,4):GOTO1000
15420 PRINT@X5+64*Y5,YY$;:RETURN
```

Normal-arrow firing routine.

```
15500 GOSUB15699
15505 X=RND(0)/2:IFRC=1THENX=X-.1
15506 IFRC=2THENX=X+.1
15507 X=X-(EP/1000)
15510 X=X-.2
15511 X=X-(DX/100)
15515 IFX>WTHENRETURN
15520 X=RND(0):IFRC=1THENX=X+.2ELSEX=X+.1
15521 IFRC=2THENX=X-.1
15525 MS=MS-X:RETURN
15599 RETURN
```

Magic-arrow firing routine.

```
15600 GOSUB 15699
15601 IFM1(RM)=8THENRETURN
15605 X=RND(0)/2:IFRC=1THENX=X-.1
15606 X=X-(DX/100)
15607 IFRC=2THENX=X+.1
15608 X=X-(EP/1000)
15610 IFX>WTHENRETURN
15620 X=RND(0):IFRC=1THENX=X+.1
15621 IFRC=2THENX=X-.1
15625 MS=MS-X:RETURN
15698 RETURN
```

Calculate monster range, aim, and shoot arrow.

```
15699 IFWX=0THENWX=31:IFWY=0THENWY=8
15700 X6=X5*2-1:Y6=Y5*3:X7=WX*2-1:Y7=WY*3
15701 IFX6=X7THENSL=SGN(Y7-Y6):X8=X7+1:X9=X6+1ELSESLP=(Y6-Y7)/(X
6-X7):IFX6>X7THENX8=X6:X9=X7+4ELSEIFX7>X6THENX8=X6+4:X9=X7
15705 IFX6=X7THENIFY7<Y6THENY7=Y7+3:Y6=Y6-2ELSEIFY7>Y6THENY7=Y7-
2:Y6=Y6+3
15706 Y2=Y7:IFABS(WY-Y5)=1THENY2=Y2+SGN(SLP)*3
15707 Y8=Y6:Y9=Y7:Y=Y8
15709 IFX6>X7THENSLP=-SLP
15710 FORX=X8TOX9+.1STEPSGN(X7-X6)
15711 IFY>47ORY<00RX>127ORX<0THENNEXT:GOTO15750
15715 IFPOINT(X,Y)=-1THENX9=X-1:GOTO15750
15720 SET(X,Y):Y=Y+SLP:IFX9<>X8THENNEXTXELSEIFY<>Y2THENNEXTX
15750 Y=Y8:FORX=X8TOX9+.1STEPSGN(X7-X6):RESET(X,Y):Y=Y+SLP:IFX9<
>X8THENNEXTXELSEIFY<>Y2THENNEXTX
15760 RETURN
```

Subroutine for close combat with a monster.

```
16000 IFABS(X5-WX)>1ORABS(Y5-WY)>1THENRETURN
16001 IFM1(RM)=8THENRETURN
16002 IFM1(RM)=7THENRETURN
16003 IFM1(RM)=6THENW=W-.05
16005 X=RND(0):IFRC=0THENX=X-.1
16006 X=X-(DX/100)
16007 IFRC=2THENX=X-.3
16008 X=X-(EP/1000)
16010 IFX>WTHENRETURN
16015 X=RND(0):IFRC=0THENX=X+.1
16016 X=X+(ST/100)
16017 IFRC=2THENX=X+.2
16020 MS=MS-X:RETURN
```

Subroutine for opening a treasure chest.

```
17000 IFABS(TX-X5)>1THENRETURN
17005 IFABS(TY-Y5)>1THENRETURN
17010 PRINT@TX+64*TY," ";
17011 TX=0:TY=0
17015 PRINT@832+41,T$(T1(RM));:FORX=1TO1000:NEXT
17020 FORX=41TO63:PRINT@832+X," ";:NEXT
17021 IFT1(RM)=10THENPT=PT+1:GOTO17026ELSEIFT1(RM)=11THENA2=A2+1
0:GOTO17026ELSEIFT1(RM)=12THENA1=A1+10:GOTO17026
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):IFM=2ORM=5ORM=7THENRETURN
18005 GOSUB15699:PRINT@WX+(64*WY)," ";:WA=WX:WB=WY:WX=X5:WY=Y5:G
OSUB16000:WX=WA:WY=WB:PRINT@X5+(64*Y5),YY$;:RETURN
```

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

```
20000 INPUT"WOULD YOU LIKE TO SEE THE TREASURES YOU RETRIEVED";A
$:IFLEFT$(A$,1)="Y"THENFORX=1TO9:PRINTT$(X),"NUMBER RETRIEVED: "
;TS(X):NEXT
20010 FORX=1TO9:TS(X)=0:NEXT
20011 A1=A1+A3:A2=A2+A4
20015 INPUT"WOULD YOU LIKE TO SAVE THIS GAME";A$:IFLEFT$(A$,1)<>
"Y"THEN20028
```

```
20018 INPUT"TO CASSETTE OR DISK ";A$:IF LEFT$(A$,1)="C"THENINPUT
"PRESS ENTER TO BEGIN SAVE ";A$:GOTO20024
20019 IFLEFT$(A$,1)<>"D"THEN20018
20020 OPEN"O",1,"QUEST/DAT"
20021 FORX=1TO58:PRINT#1,M1(X);M2(X);T1(X):NEXT
20022 CLOSE:GOTO20027
20024 FORX=1TO58:PRINT#-1,M1(X),M2(X),T1(X):NEXT
20027 PRINT"SAVE COMPLETE"
20028 INPUT"WOULD YOU LIKE TO STOP NOW";A$:IFLEFT$(A$,1)<>"Y"THE
NRETURN
20030 PRINT"OK. SO THAT YOU CAN USE THIS CHARACTER AGAIN:"
20035 PRINT"NAME: ";NM$;" RACE: ";:IFRC=0THENPRINT"HUMAN":ELSEIF
RC=1THENPRINT"ELF":ELSEPRINT"DWARF"
20040 PRINT"WOUNDS: ";W#100;"%"
20045 PRINT"HEALING POTIONS: ";PT;"  HOLY WATER: ";HW
20050 PRINT"ARROWS: ";A1;"  MAGIC ARROWS: ";A2
20055 PRINT"GOLD: ";GP;"  EXPERIENCE: ";EP
20060 PRINT"STRENGTH: ";ST;"  DEXTERITY:  ";DX
20065 INPUT"WOULD YOU LIKE TO TRY AGAIN AS A *NEW* CHARACTER";A$
:IFLEFT$(A$,1)="Y"THENRUN
20099 PRINT:PRINT"COME QUESTING AGAIN SOME TIME!!!":END
```

Subroutine to create new characters.

```
21000 PRINT"OK, I'LL MAKE YOU ONE.":FORX=1TO1000:NEXT
21005 GP=RND(20)+5:ST=RND(17)+3:DX=RND(17)+3:RC=RND(3)-1:A1=3:A2
=RND(10):PT=RND(3)+1:HW=RND(5):EP=0:W=1
21010 PRINT"STRENGTH: ";ST;"  DEXTERITY: ";DX
21015 PRINT"GOLD: ";GP;"  HEALING POTIONS: ";PT
21020 PRINT"HOLY WATER: ";HW;"  RACE: ";:IFRC=1THENPRINT"ELF"ELS
EIFRC=2PRINT"DWARF"ELSEPRINT"HUMAN"
21025 PRINT"ARROWS: ";A1;"  MAGIC ARROWS: ";A2
21027 IFTRS80MODEL=3THENINPUT"IS THIS CHARACTER FEMALE";A$:IFLEF
T$(A$,1)="Y"THENYY$=CHR$(254)
21028 IFTRS80MODEL=3THENPOKE16409,0
21030 INPUT"WHAT WILL YOU NAME THIS CHARACTER";NM$:PRINT"HAVE A
FUN QUEST, ";NM$;"!!!":FORX=1TO1000:NEXT:CLS
21035 IFTRS80MODEL=3THENPOKE16409,1
21040 RETURN
```

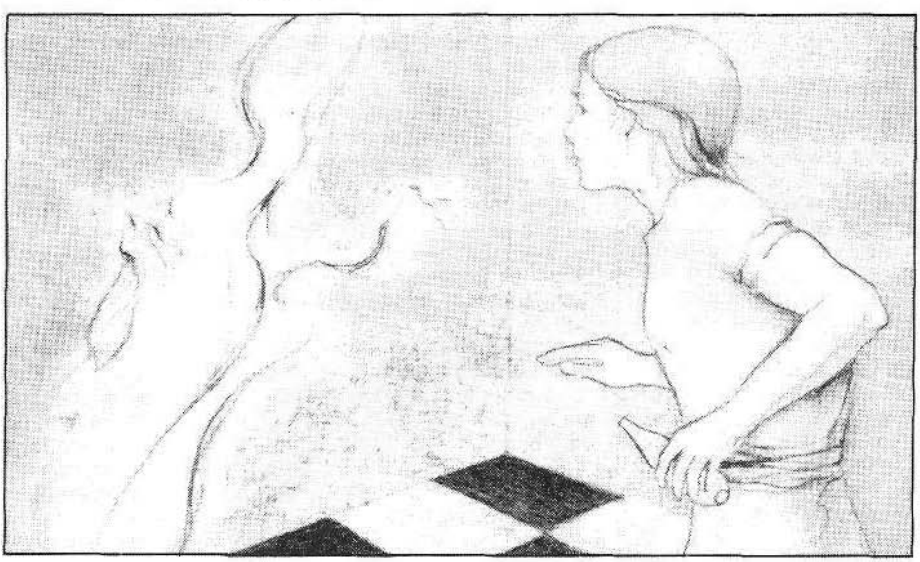
Error-handling routine for arrow shooting routines.

```
29999 END:REM PROTECT ERROR ROUTINE FOR ARROW SHOOTING
30000 IFERL=15750THENRESUME15760ELSEIFERL=4096THENRESUMENEXTELSE
PRINTERL,ERR/2+1:FORX=1TO1000:NEXT:RESUMENEXT
```

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS          TRS-80 BASIC       SS
SS      'Quest 2 Database'     SS
SS   Author:  Tigre Wenrich    SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
2 PRINT@476,"QUEST 2";:IFTRS80MODEL=3THENPOKE16420,1
3 A$="                  * * * * * * * * * *  Q U E S T  2  W
 A S  W R I T T E N  B Y  B R I A N  R E Y N O L D S  * * *
 * * * * * *  (HAVE FUN!)   "'QUEST2 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
```



TRS-80® SWAT TABLE FOR:
QUEST 1

LINES	SWAT CODE	LENGTH
1 - 100	LI	650
105 - 160	NL	484
165 - 220	XD	290
225 - 340	HZ	292
350 - 460	QT	295
470 - 580	ZM	292
600 - 812	YT	428
813 - 850	OY	499
860 - 905	YS	500
906 - 940	RG	534
941 - 991	ZJ	457
992 - 1010	PI	517
1015 - 1065	IO	441
1070 - 1110	MJ	508
1115 - 1215	PB	406
1220 - 5020	HA	527
10000 - 11005	PM	502
11010 - 11050	GR	528
11055 - 15305	FT	405
15310 - 15510	CU	317
15511 - 15608	WJ	204
15610 - 15709	CB	378
15710 - 16006	GS	325
16007 - 17015	YJ	225
17020 - 20015	OW	541
20018 - 20045	OO	489
20050 - 21020	WI	522
21025 - 30000	ZO	371

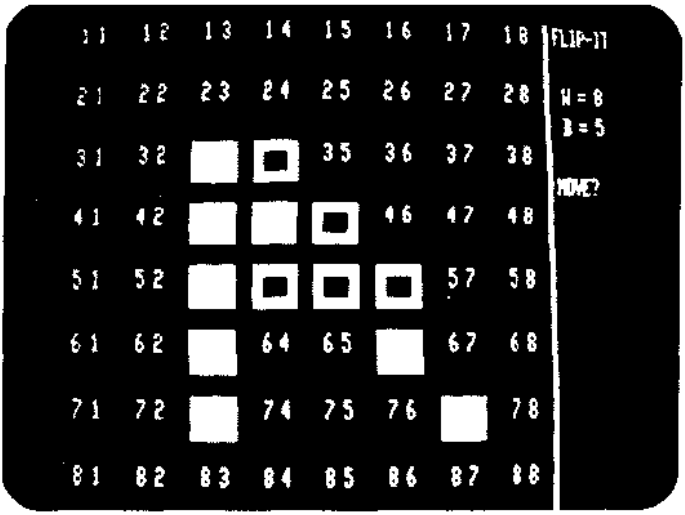
NOTES:

TRS-80® SWAT TABLE FOR:
QUEST 2 DATABASE

(Modified Parameters: NU = 2 B = 300)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
2 - 3	RD	240	290 - 300	DG	49
115 - 120	GF	48	310 - 320	DY	49
125 - 130	ZH	44	330 - 340	BC	47
135 - 140	CR	45	350 - 360	JQ	50
145 - 150	BK	44	370 - 380	FE	47
155 - 160	GA	46	390 - 400	KS	49
165 - 170	JU	48	410 - 420	JG	48
175 - 180	LE	48	430 - 440	LW	48
185 - 190	NX	49	450 - 460	QB	49
195 - 200	QT	49	470 - 480	TD	50
205 - 210	PQ	49	490 - 500	SM	49
215 - 220	PK	49	510 - 520	HQ	48
225 - 240	RH	49	530 - 540	CE	49
250 - 260	HP	48	550 - 560	DI	49
270 - 280	DT	50	570 - 580	DI	48

NOTES:



FLIP-IT

by Michael Prescott

TRS-80 version by Alan J. Zett
***Flip-It* is a TRS-80 program requiring 16K 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.

Each square on the board is numbered, so all you have to do is type the number of the square you want. 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. 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$: General purpose.
B(*): Array representing board layout. Contents: 9 represents off-board; 0 is unoccupied square; 1 is white; -1 is black.
B$: Graphics string for border.
BL: Black piece value (-1).
BP: Number of black pieces on the board.
C: Value (-1 or 1) of the color calling the "legal check" routine.
F: At start of game, F = 1 if you choose to go first. During the game, used as a flag to flip pieces.
FL: If FL = 0, legal check will not flip pieces.
IP: "I pass" flag for computer.
LG: LG = 1 indicates a legal move.
OP: Color of computer's chips.
OP$: Graphics string for the computer's pieces.
P:Temporary holding variable for PDL.
PDL: Position of a square under consideration.
P2: Number of chips on board.
RF: Return flag. If = 1, then RETURN from a normal (non-GOSUB) routine.
SC: Color being searched for by "legal check."
X,Y,Z: Miscellaneous.
YO: Player's color.
YO$: Graphics string for player's pieces.
YP: "You pass" flag.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        TRS-80 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.

```
90 CLEAR200:DEFINTA-Z:DIMA(60),B(99):A$="11811888316113831686386
8336336664151148415854858435334643565465642522474257547573262237
3267637672171128217872878227227777":Z=1
```

```
100 FORX=1TO120STEP2:A(Z)=ASC(MID$(A$,X,1))-48+(ASC(MID$(A$,X+1,
1))-48)*10:Z=Z+1:NEXT:FORX=0TO99:B(X)=0:NEXT:B(45)=-1:B(54)=-1:B
(44)=1:B(55)=1:GOSUB620
110 PDL=0:B=0:FL=0:P2=0:RF=0:BL=-1:WH=1:OP=0:YO=0:FORX=0TO9:B(X)
=9:B(X+90)=9:NEXT:FORX=9TO89STEP10:B(X)=9:B(X+1)=9:NEXT
120 B$=CHR$(191)+STRING$(3,131)+CHR$(191)+STRING$(5,24)+CHR$(26)
+STRING$(5,131):W$=STRING$(5,191)+STRING$(5,24)+CHR$(26)+STRING$
(5,131)
```

Get choice of color, who goes first, and whether to use a normal board. Set up pieces.

```
130 CLS:F=0:A$="":PRINT@24,"F L I P - I T":PRINT:INPUT"WHICH COL
OR DO YOU WANT (W OR B) ";A$:IFLEFT$(A$,1)<>"W"THEN150
140 YO$=W$:OP$=B$:YO=WH:OP=BL:GOTO160
150 YO$=B$:OP$=W$:YO=BL:OP=WH
160 A$="":INPUT"DO YOU WANT TO GO FIRST ";A$:IFLEFT$(A$,1)="Y"TH
EN F=1
170 A$="":INPUT"WANT TO SET UP YOUR OWN GAME ";A$:CLS:GOSUB180:I
FLEFT$(A$,1)="Y"THENB(44)=0:B(45)=0:B(54)=0:B(55)=0:GOTO280ELSEI
FF=1THEN340ELSEGOSUB190:GOTO410
```

Draw game board.

```
180 FORX=0TO7:FORY=0TO7:PRINT@X*128+Y*7,STR$(X+1);STR$(Y+1);:NEX
TY:NEXTX:RETURN
```

Plot pieces and update game display data.

```
190 P2=0:WP=0:BP=0:FORX=0TO99:IFB(X)<>0ANDB(X)<>99THENY=INT((X-1
1)/10)*128+((X-11)-INT((X-11)/10)*10)*7
200 IFB(X)=1THENPRINT@Y,W$;:WP=WP+1:P2=P2+1
210 IFB(X)=-1THENPRINT@Y,B$;:BP=BP+1:P2=P2+1
220 NEXT:FORY=0TO47:SET(110,Y):NEXT:PRINT@56,"FLIP-IT";:PRINT@18
5,"W =";WP;:PRINT@249,"B =";BP;:RETURN
```

No-possible-move command.

```
230 PRINT@376,"CHECKING";:FORZ=1TO60:PDL=A(Z):P=PDL:C=YO:SC=OP:F
L=0:GOSUB500:IFLG=1THEN240ELSENEXT:IFIP=1THEN430ELSEYP=1:GOTO410
240 PRINT@376,"?WHAT    ";:PRINT@440,"ABOUT  ";:A$=LEFT$(STR$(PDL
),2)+" "+RIGHT$(STR$(PDL),1):FORX=1TO15:PRINT@504,A$;:FORY=0TO99
:NEXT:PRINT@504,"        ";:FORY=0TO99:NEXT:NEXT:GOTO350
```

Find player's best move.

```
250 PRINT@376,"LOOKING ";:FORZ=1TO60:PDL=A(Z):P=PDL:C=YO:SC=OP:F
L=0:GOSUB500:IFLG=1THEN260ELSENEXT:IFIP=1THEN430ELSEYP=1:GOTO410
260 PRINT@440,"LOOKS   ";:PRINT@504,"GOOD!";:A$=LEFT$(STR$(PDL),
2)+" "+RIGHT$(STR$(PDL),1):FORX=1TO15:PRINT@376,A$;:FORY=0TO99:N
EXT:PRINT@376,"        ";:FORY=0TO99:NEXT:NEXT:PRINT@504,"      ";
:GOTO350
```

Quit-game routine.

```
270 A$="":CLS:PRINT@24,"F L I P - I T":PRINT:INPUT"DO YOU REALLY
 WANT TO QUIT ";A$:IFLEFT$(A$,1)<>"Y"THENCLS:GOSUB180:GOTO340ELS
E440
```

Set up your own board.

```
280 RF=1:PRINT@504,"(D=DONE)";:GOSUB350:PRINT@376,"COLOR?  ";
290 A$=INKEY$:IFA$=""THEN290ELSEIFA$="D"THENPRINT@504,"        "
;:GOSUB180:IFF=1THEN340ELSEGOSUB190:GOTO410
300 PRINT@INT((PDL-11)/10)*128+((PDL-11)-INT((PDL-11)/10)*10)*7,
;
310 IFA$="W"THENPRINTW$;:B(PDL)=WH
320 IFA$="B"THENPRINTB$;:B(PDL)=BL
330 GOTO280
```

Get player's move.

```
340 IFP2=64THEN430ELSEGOSUB190:IFP2=64THEN430
350 PDL=0:YP=0:PRINT@440,"       ";:PRINT@376,;:IFRF=1THENPRINT"
SQUARE? ";ELSEPRINT"MOVE?   ";
360 FORX=0TO1
370 A$=INKEY$:IFA$=""THEN370ELSEIFA$="R"THENCLS:GOSUB180:GOSUB19
0:GOTO350ELSEIFA$="Q"THEN270ELSEIFA$="N"THEN230ELSEIFA$="B"THEN2
50ELSEIFA$="D"ANDRF=1THENRF=0:RETURN
380 IFA$<"1"ORA$>"8"THENPRINT@440,"INVALID";:FORY=1TO555:NEXT:GO
TO350
390 PRINT@441+X*2,A$;:PDL=PDL+VAL(A$)*10^(1-X):NEXT:IFRF=1THENRF
=0:RETURN
400 C=YO:SC=OP:FL=1:GOSUB500:IFLG=0THENA$="0":GOTO390ELSEGOSUB19
0
```

Choose computer's move.

```
410 IP=0:PRINT@376,"MAYBE   ";:FORZ=1TO60:PDL=A(Z):PRINT@440,LEF
T$(STR$(PDL),2)" "RIGHT$(STR$(PDL),1);:P=PDL:IF B(P)<>0THENNEXTZ
```

```
ELSEC=OP:SC=YO:FL=1:GOSUB500:IFLG=0THENNEXTZELSE340
```

Computer must pass.

```
420 IF P2<>64 AND YP=0 THEN PRINT@569,"PASS!";:IP=1:FORX=0TO555:
NEXTX:PRINT@569,"     ";:GOTO340
```

End-game routine.

```
430 PRINT@568,"NO MOVES";:PRINT@634,"LEFT!";:PRINT@760,"* GAME *
";:PRINT@824,"* OVER *";:FORX=0TO5000:NEXTX
440 CLS:PRINT@24,"F L I P - I T":PRINT:PRINT"FINAL SCORE:":PRINT
:PRINT"    WHITE ="WP:PRINT"    BLACK ="BP:PRINT
450 IFWP=BPTHENPRINT"IT'S A DRAW! I'LL BEAT YOU NEXT TIME!!":PRI
NT:GOTO480ELSEIF(WP>BPANDOP=WH)OR(BP>WPANDOP=BL)THENPRINT"I ONCE
 AGAIN PROVE MY SUPERIORITY! I WON BY";:GOTO470
460 PRINT"YOU WON THIS ROUND BY";
470 PRINTABS(WP-BP)"CHIPS!":PRINT:PRINT
```

Check for new game.

```
480 A$="":INPUT"WANT TO PLAY AGAIN ";A$:IFLEFT$(A$,1)="Y"RUN
490 CLEAR50:PRINT:PRINT"OK. SEE YOU LATER!":END
```

Legality check: combines a search for legal moves with a routine to flip pieces.

```
500 LG=0::IFB(PDL)<>0THEN610
510 FORY=1TO8:F=0:ONYGOTO520,530,540,550,560,570,580,590
520 P=PDL:FORX=1TO8:P=P+10:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CAND
X=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO660
0ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOT
O520
530 P=PDL:FORX=1TO8:P=P-10:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX
=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600
ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO
530
540 P=PDL:FORX=1TO8:P=P+1:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CANDX
=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600
ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO
540
550 P=PDL:FORX=1TO8:P=P-1:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX=
1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600E
LSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO5
50
```

```
560 P=PDL::FORX=1TO8:P=P+9:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CAND
X=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO60
0ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOT
0560
570 P=PDL:FORX=1TO8:P=P-9:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX=
1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600E
LSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO5
70
580 P=PDL:FORX=1TO8:P=P+11:IFP>99THEN600ELSEIFB(P)=COR(B(P)=CAND
X=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO60
0ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOT
0580
590 P=PDL:FORX=1TO8:P=P-11:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX
=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600
ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO
590
600 FORX=0TO1:NEXTY:NEXTY
610 FORX=0TO1:NEXTX:FORY=0TO1:NEXTY:RETURN
```

Instructions.

```
620 CLS:A$="":PRINT@512,;:INPUT"NEED INSTRUCTIONS ";A$:IFLEFT$(A
$,1)<>"Y"THEN110
630 CLS:PRINT@24,"F L I P - I T":PRINT:PRINT"THE OBJECT OF THIS
GAME IS TO COMPLETELY FILL THE BOARD WITH AS MANY PIECES OF YOUR
 COLOR AS YOU CAN. TO DO THIS YOU MUST OUT-  FLANK YOUR OPPONENT
'S PIECES AND FLIP THEM TO YOUR COLOR."
640 PRINT:PRINT"OUTFLANKING CAN OCCUR HORIZONTALLY, VERTICALLY,
OR DIAGONALLY.  THE GAME ENDS WHEN THE BOARD IS FULL, OR WHEN BO
TH PLAYERS CAN'TMOVE. AT THAT TIME, WHOEVER HAS THE MOST PIECES
WINS."
650 PRINT@977,"(PRESS <ENTER> TO CONTINUE)";
660 IFINKEY$<>CHR$(13)THEN660
670 CLS:PRINT@24,"F L I P - I T":PRINT:PRINT"YOU MAKE MOVES BY E
NTERING THE ROW NUMBER COMBINED WITH THE     COLUMN NUMBER. FOR
EXAMPLE: THE UPPER RIGHT-HAND CORNER IS THE  POSITION 18 - THAT
IS 1 FOR THE FIRST ROW AND 8 FOR THE EIGHTH"
680 PRINT"COLUMN. IF YOU MAKE A MISTAKE ENTERING A NUMBER, TYPE
A NINE FORANY OF THE NUMBERS AND IT WILL MARK THE ENTRY AS INVAL
ID.":PRINT"DURING THE GAME YOU ALSO HAVE THE FOLLOWING OPTIONS:"
:PRINT
690 PRINT"Q - TO QUIT THE GAME":PRINT:PRINT"N - NOT ABLE TO MOVE
":PRINT:PRINT"B - ASK FOR THE BEST MOVE":PRINT@977,"(PRESS <ENTE
R> TO START GAME)";
700 IFINKEY$<>CHR$(13)THEN700ELSE110
```

TRS-80® SWAT TABLE FOR:
FLIP-IT

LINES	SWAT CODE	LENGTH
90 - 130	YX	570
140 - 220	AL	540
230 - 270	RD	577
280 - 380	VL	517
390 - 450	XF	623
460 - 530	FK	504
540 - 570	IS	575
580 - 630	AN	613
640 - 680	NM	660
690 - 700	HR	147

NOTES:

Battlefield

by Joe Humphrey

TRS-80 version by Jon R. Voskuil
***Battlefield* is a TRS-80 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 in-

creased, 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$).
BL: Index to black player (=0).
BS$: Backspace character, CHR$(8).
CC: Cursor column.
CH$: A character.
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, "#".
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: PRINT@ position for number of forces.

HC: Half of NC.
IPL: Index to a player.
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        TRS-80 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 60
19  '----- PRINT CONTENTS OF POSITION (COL,ROW) TO SCREEN -----
20 CP=CP(COL,ROW): PRINT@ PA%(COL,ROW),BC$; P$(CP);: NB=SF(BL,CO
L,ROW): NW=SF(WH,COL,ROW): SF=NB+NW: IF SF(1-CP,COL,ROW)=0 THEN
30
25 PRINT CHR$(NB/SF*25+65); CHR$(NW/SF*25+65);: GOTO 35
30 PRINT RIGHT$(STR$(SF),2);
35 PRINT P$(CP); BC$;: RETURN
58 REM
59 REM----- MAIN PROGRAM CONTROL ROUTINE -----
60 CLEAR 200
```

```
65 B$=CHR$(138): W$="-"
70 NC=8: NR=10
80 GOSUB 1000
90 GOSUB 2000
100 GOSUB 3000
110 GOSUB 4000
120 GOSUB 5000
130 GOSUB 6000
140 GOSUB 7000
150 GOTO 8000
160 CLS: END
998 '
999 '----- PRINT INSTRUCTIONS -----
1000 CLS
1010 PRINT "BATTLEFIELD: A TERRITORIAL GAME                    BY J
OE HUMPHREY"
1020 PRINT TAB(15) "(S-80 TRANSLATION BY JON VOSKUIL)": PRINT
1040 PRINT"    TWO PLAYERS CONTROL FORCES ON A";NC;"BY";NR;"PLAY
ING FIELD."
1050 PRINT"SQUARES OWNED BY ONE SIDE ARE SHOWN WITH THE NUMBER O
F FORCES","BETWEEN VERTICAL BARS (";CHR$(138);"  ";CHR$(138);");
 SQUARES OWNED BY THE OTHER SIDE"
1060 PRINT"ARE SHOWN WITH THE NUMBER OF FORCES BETWEEN HYPHENS (
-  -).","    SQUARES CONTAINING FORCES OF BOTH SIDES ARE SHOWN W
ITH"
1070 PRINT"LETTERS REPRESENTING THE RELATIVE FORCE SIZES; AT THE
 END OF","EACH TURN, BATTLES ARE FOUGHT IN THESE SQUARES TO DETE
RMINE"
1080 PRINT"WHO OWNS THEM.  THE FIRST PLAYER TO OWN ALL THE SQUAR
ES WINS."
1090 PRINT"    AT THE BEGINNING OF EACH ROUND, NEW FORCES ARE AD
DED TO EACHSIDE, AND EACH SIDE'S MOVEMENT ALLOWANCE IS INCREASED
."
1110 RETURN
1998 '
1999 '----- INITIALIZE VARIABLES -----
2000 BL=0: WH=1
2010 F=0: T=1
2020 MC=NC-1: MR=NR-1
2025 DIM PA%(MC,MR): FOR COL=0 TO MC: FOR ROW=0 TO MR: PA%(COL,R
OW)= ROW*64 + COL*16 + (10-NC)*3 +1: NEXT: NEXT
2030 DIM DC(1),EC(1),LC(1),PC(1)
2040 DIM MA(1),MF(1),NS(1),PN$(1),TF(1)
2050 DIM CP(MC,MR),SF(1,MC,MR)
2060 LC(BL)=1: DC(BL)=16: EC(BL)=21
```

```
2070 LC(WH)=LC(BL)+20: DC(WH)=DC(BL)+20: EC(WH)=EC(BL)+19
2080 FR=839: SR=855: AR=914
2090 BS$=CHR$(8)
2100 SP$=" ": CU$="#"
2110 DIM P$(1): P$(BL)="-": P$(WH)=CHR$(138)
2120 HC=INT(NC/2)
2130 FOR ROW=0 TO MR: FOR COL=0 TO HC-1: CP(COL,ROW)=BL: NEXT: F
OR COL=HC TO MC: CP(COL,ROW)=WH: NEXT: NEXT
2170 NS(BL)=NR*HC: NS(WH)=NR*NC-NS(BL)
2190 RETURN
2998 '
2999 '----- INPUT PLAYERS' NAMES -----
3000 PRINT @900,"PLAYER 1'S NAME: ";: INPUT PN$
3010 IF PN$="" THEN 3000
3020 IF LEN(PN$)>10 THEN PN$=LEFT$(PN$,10)
3025 PN$(BL)=PN$: PN$=""
3030 PRINT @900, STRING$(63,32);: PRINT @900, "PLAYER 2'S NAME:
";: INPUT PN$
3040 IF PN$="" THEN 3030
3050 IF LEN(PN$)>10 THEN PN$=LEFT$(PN$,10)
3055 PN$(WH)=PN$
3060 RETURN
3998 '
3999 '----- DRAW PLAYING FIELD AND DISPLAY PLAYER DATA -----
4000 BC$=SP$
4010 CLS
4020 FOR ROW=0 TO MR: FOR COL=0 TO MC: GOSUB 20: NEXT: NEXT
4050 PRINT@ 704, STRING$(64,176);: FOR I=0 TO 128 STEP 64: PRINT
@ 799+I, CHR$(149);: NEXT I
4060 FOR IPL=BL TO WH: PC=IPL*32
4065 L=14-INT(LEN(PN$(IPL))/2)
4070 PRINT@ 769+PC, STRING$(L,95); PN$(IPL); STRING$(L,95);
4080 PRINT@ 832+PC, "FORCES:        SQUARES:";
4090 PRINT@ 900+PC, "MOVEMENT LEFT:";
4100 PRINT@ FR+PC, TF(IPL);
4110 PRINT@ SR+PC, NS(IPL);
4120 PRINT@ AR+PC, MA(IPL);
4130 NEXT IPL: RETURN
4998 '
4999 '----- BEGIN BLACK'S TURN.  ADD FORCES & INCREASE MA'S ----
5000 BC$=SP$
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 STEP MC
```

```
5050 CP=CP(COL,ROW): SF=SF(CP,COL,ROW)
5060 NF=MF(CP): IF SF+NF>99 THEN NF=99-SF
5070 IF NF>0 THEN SF(CP,COL,ROW)=SF+NF: GOSUB 20
5080 TF(CP)=TF(CP)+NF: MA(CP)=MA(CP)+MF(CP)
5090 NEXT COL,ROW
5100 FOR IPL=BL TO WH
5110 PRINT@ FR+IPL*32, TF(IPL);"  ";
5120 PRINT@ AR+IPL*32, MA(IPL);"  ";
5130 NEXT IPL: PL=BL: CC=0
5140 RETURN
5998 '
5999 '----- EXECUTE PLAYER'S MOVES TILL DONE OR MA DEPLETED ----
6000 CR=INT(NR/2)
6010 NG=0
6020 PF=F
6030 GOSUB 14000
6040 BC$=CU$: COL=CC: ROW=CR: GOSUB 20
6050 GOSUB 11000
6060 MG=SF(PL,CC,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 6060
6120 BC$=SP$: COL=CC: ROW=CR: GOSUB 20
6130 NG=0:GOSUB 11000
6140 PRINT@ 960, STRING$(63,32);
6160 RETURN
6998 '
6999 '----- FIGHT BATTLES TO DEATH IN DISPUTED SQUARES -----
7000 IF NS(PL)=0 THEN 7170
7010 FOR ROW=0 TO MR: FOR COL=0 TO MC
7020 CP=CP(COL,ROW): IF CP=PL OR SF(PL,COL,ROW)=0 THEN 7160
7030 NS(CP)=NS(CP)-1
7040 NB=SF(BL,COL,ROW): NW=SF(WH,COL,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(0) THEN NB=NB-1: TF(BL)=TF(BL)-1
7090 IF NW<SF*RND(0) THEN NW=NW-1: TF(WH)=TF(WH)-1
7100 FOR ZP=BL TO WH: PRINT@ FR+32*ZP, TF(ZP);: NEXT ZP
7110 SF(BL,COL,ROW)=NB: SF(WH,COL,ROW)=NW
7120 GOSUB 20: GOTO 7050
7130 GOSUB 20
7140 CP=CP(COL,ROW): NS(CP)=NS(CP)+1
```

```
7150 FOR ZP=BL TO WH: PRINT@ SR+32*ZP, NS(ZP);: NEXT ZP
7160 NEXT: NEXT
7170 RETURN
7998 '
7999 '----- END OF TURN; CHECK FOR END OF GAME -----
8000 IF NS(BL)*NS(WH)>0 THEN 8100
8020 WP$=PN$(BL): IF NS(BL)=0 THEN WP$=PN$(WH)
8025 FOR I=LEN(WP$)-1 TO 1 STEP -1: WP$=LEFT$(WP$,I) + " " + RIG
HT$(WP$,LEN(WP$)-I): NEXT I
8030 FOR Z=1 TO 30: PRINT@ 960, STRING$(63,32);: FOR ZZ=1 TO 10:
 NEXT ZZ
8040 PRINT@ 960, WP$;"   W I N S ! ! !";: FOR ZZ=1 TO 15: NEXT Z
Z: NEXT Z
8050 PRINT@ 990,"ANOTHER GAME? (Y/N)";
8060 CH$=INKEY$: IF CH$="" THEN 8060
8070 IF CH$="N" THEN 8130
8090 GOTO 60
8100 IF PL=WH THEN 120
8110 PL=WH: CC=MC
8120 GOTO 130
8130 GOTO 160
9998 '
9999 '----- INPUT & EXECUTE A COMMAND -----
10000 CH$=INKEY$: IF CH$="" THEN 10000
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$): GOTO 10160
10070 IF CH$=BS$ THEN NG=INT(NG/10): PF=1: GOTO 10160
10080 IF CH$="E" THEN 10160
10090 IF CH$="H" THEN GOSUB 12000: GOSUB 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 10160
10150 GOSUB 13000
10160 RETURN
10998 '
10999 '---- ADJUST NO. OF FORCES BEING MOVED, UPDATE SCREEN -----
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 PRINT@ 982,;
11040 IF NG=0 THEN PRINT "<TYPE H FOR HELP>";: GOTO 11080
```

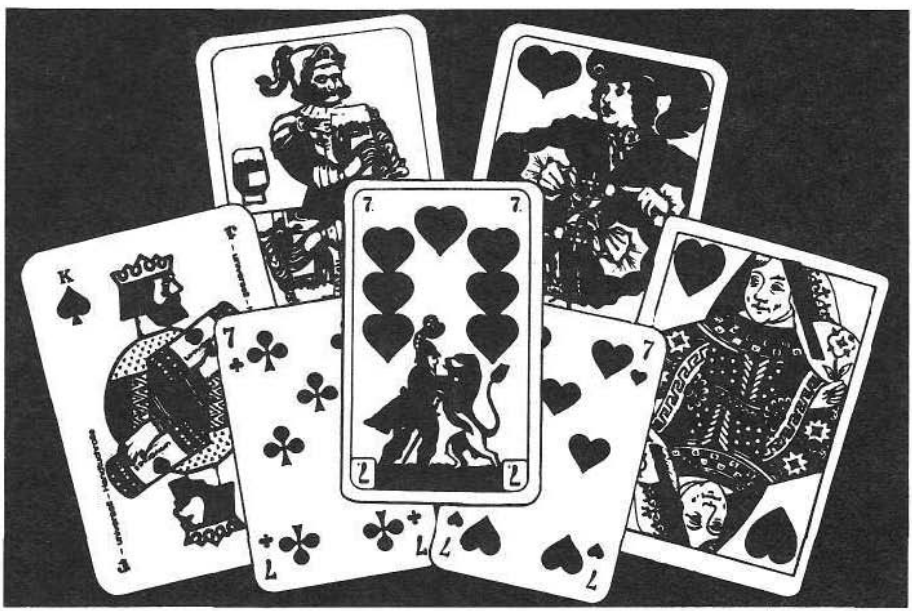
```
11050 PRINT "<MOVING";NG;"FORCE";
11060 IF NG>1 THEN PRINT "S";
11070 PRINT">";
11080 PRINT "    ";
11090 RETURN
11998 '
11999 '---- PRINT LIST OF COMMANDS IN RESPONSE TO "H" -----
12000 CLS: PRINT@ 64, "COMMAND:", "DESCRIPTION:"
12005 PRINT STRING$(8,131),STRING$(12,131)
12010 PRINT: PRINT "A NUMBER", "THE NUMBER OF FORCES TO MOVE (E.
G., 23)."
12020 PRINT "   ";CHR$(93), "DELETE LAST DIGIT OF NUMBER."
12030 PRINT"   E", "END TURN.  (TURN ALSO ENDS WHEN MOVEMENT",,
"   ALLOWANCE IS ZERO.)"
12040 PRINT "   H", "PRINT THIS COMMAND LIST."
12050 PRINT "   I", "MOVE SOME FORCES UP."
12060 PRINT "   J", "MOVE SOME FORCES LEFT."
12070 PRINT "   K", "MOVE SOME FORCES RIGHT."
12080 PRINT "   M", "MOVE SOME FORCES DOWN."
12090 PRINT "   S", "SURRENDER TO THE OTHER SIDE."
12100 PRINT@ 980, "<HIT ANY KEY TO RETURN TO GAME>";
12110 CH$=INKEY$: IF CH$="" THEN 12110
12120 CH$="H": RETURN
12998 '
12999 '---- MOVE CURSOR & FORCES, UPDATE SCREEN -----
13000 SF(PL,CC,CR)=SF(PL,CC,CR)-NG
13010 BC$=SP$: COL=CC: ROW=CR: GOSUB 20
13020 CC=CC+DC: CC=CC-(CC<0)+(CC>MC)
13030 CR=CR+DR: CR=CR-(CR<0)+(CR>MR)
13040 FS=SF(FL,CC,CR): IF FS+NG>99 THEN SF(PL,COL,ROW)=SF(PL,COL
,ROW)+NG-(99-FS): NG=99-FS: GOSUB 20
13050 SF(PL,CC,CR)=FS+NG
13060 IF CC=COL AND CR=ROW THEN NG=0
13070 BC$=CU$: COL=CC: ROW=CR: GOSUB 20
13080 MA(PL)=MA(PL)-NG: GOSUB 11000
13100 PRINT@ AR+PL*32, MA(PL);
13120 RETURN
13998 '
13999 '---- FLASH PLAYER'S NAME WHOSE MOVE IS BEGINNING -----
14000 PRINT@ 960, STRING$(63,32);
14010 FOR Z=1 TO 4: PRINT@ 960+45*PL, STRING$(18,32);
14020 FOR ZZ=1 TO 15: NEXT ZZ
14030 PRINT@ 960+45*PL, PN$(PL);"'S TURN";
14040 FOR ZZ=1 TO 20: NEXT ZZ
14050 NEXT Z: RETURN
```

TRS-80® SWAT TABLE FOR:
BATTLEFIELD

LINES	SWAT CODE	LENGTH
10 - 80	XI	395
90 - 1010	PE	210
1020 - 1070	SM	530
1080 - 2050	GR	489
2060 - 2999	UB	388
3000 - 4000	VJ	330
4010 - 4130	KN	369
4998 - 5090	QN	350
5100 - 6040	ZY	264
6050 - 6998	JK	233
6999 - 7100	RA	433
7110 - 8025	JV	365
8030 - 9999	KG	312
10000 - 10130	KL	367
10140 - 11060	PF	306
11070 - 12050	WD	439
12060 - 13020	JK	417
13030 - 14010	HL	404
14020 - 14050	UR	92

NOTES:

Solitaire



by Larry Williams
TRS-80 version by Rich Bouchard.
***Solitaire* is a graphic card game for a TRS-80 with 16K RAM.**

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 you run the game, 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.
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        TRS-80 BASIC         SS
SS         'Solitaire'         SS
SS   Author: Larry Williams    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
```

Skip to main routines after some initialization.

```
10 CLS:CLEAR1000:DEFINTA-Z:GOTO 1110
```

Subroutine to draw cards.

```
20 PRINT@X%8+Y%64,G$(6);
```

```
30 PRINT@X$8+Y$64,MID$(VA$,VA,1);
50 PRINTG$(SU);
60 RETURN
```

Subroutine to get SU and VA.

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

Subroutine for next card.

```
100 IF HF<>0 THEN GOSUB1480:RETURN
110 IF IN>51 THEN 1490
120 OD(IN(7))=D(IN):IN=IN+1:X=7:Y=12:NU=OD(IN(7)):GOSUB70:GOSUB2
0:IN(7)=IN(7)+1
130 IF CU=7 THEN GOSUB 150
140 RETURN
```

Subroutine to draw cursor.

```
150 IF OC<>7 THEN PRINT@898+OC*8,"   "; ELSE IF IN(7)<>0 THEN PR
INT@952,CHR$(191); ELSE PRINT@952," ";
160 IF CU<>7 THEN PRINT@898+CU*8,G$(5); ELSE PRINT@952,CHR$(81);
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<>0 THEN GOSUB 1480:RETURN
```

```
290 ST=CU
300 IF IN(CU)=0 THEN GOSUB 1510:RETURN
310 IF CU=7 THEN NU=OD(IN(7)-1):GOTO 330
320 NU=C(CU,0)
330 HF=1
340 X=CU:J=0:IF CU=7 THEN J=12:PRINT@X*8+768,G$(6);:GOTO 380
350 IF P(CU,0)<>0 THEN PRINT@X*8,G$(6);:GOTO 380
360 FORY=0TO2:PRINT@Y*64+X*8,STRING$(8,32);
370 NEXTY
380 IF CU=7 THEN GOSUB 150:RETURN
385 IF IN(CU)=1 THEN RETURN
390 J=IN(CU)+2:IF J>15 THEN J=15
395 FOR I=3 TO J:PRINT@I*64+X*8,STRING$(8,32);:NEXT I
400 RETURN
```

Main subroutine to drop card.

```
410 IF HF=0 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 NU=C(CU,IN(CU)-1)
460 GOSUB70:TS=SU:TV=VA
470 IF ST=7 THEN NU=OD(IN(7)-1):GOTO 490
480 NU=C(ST,0)
490 GOSUB70:IF((TS=1)OR(TS=2))AND((SU=1)OR(SU=2))THENGOSUB1530:R
ETURN
500 IF((TS=3)OR(TS=4))AND((SU=3)OR(SU=4))THENGOSUB1540:RETURN
510 IF TV<>VA+1 THEN GOSUB1550:RETURN
515 IF VA=1 THEN GOSUB 1500:RETURN
520 IF ST=7 THEN GOSUB700:RETURN
530 FOR I=0 TO IN(ST)-1:NU=C(ST,I):C(CU,IN(CU))=NU:GOSUB70:X=CU:
Y=IN(CU):GOSUB20:IN(CU)=IN(CU)+1:C(ST,I)=0:NEXT I
540 IN(ST)=0:HF=0
550 IF P(ST,0)=0 THEN RETURN
560 NU=P(ST,0):GOSUB70:X=ST:Y=0:GOSUB20:C(ST,IN(ST))=NU:IN(ST)=1
570 FORI=0TO4:P(ST,I)=P(ST,I+1):NEXTI:P(ST,5)=0
580 RETURN
```

Drop card back on open deck.

```
590 IF ST<>7 THEN GOSUB 1560:RETURN
600 NU=OD(IN(7)-1):GOSUB70:X=CU:Y=12:GOSUB20:GOSUB150
610 HF=0
620 RETURN
```

Drop king on open space.

```
630 IF ST=7 THEN NU=OD(IN(7)-1):GOTO 650
640 NU=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=CU:Y=IN(CU):C(CU,IN(CU))=NU:GOSUB20:IN(CU)=IN(CU)+1
710 IN(7)=IN(7)-1:OD(IN(7))=0:HF=0
720 IF IN(7)=0 THEN FOR Y=12 TO 14:PRINT@Y*64+56,CHR$(30);:NEXT
Y:RETURN
730 NU=OD(IN(7)-1):GOSUB70:X=7:Y=12:GOSUB20
740 RETURN
```

Drop cards back where you got them.

```
750 FOR I=0 TO IN(CU)-1:NU=C(CU,I):GOSUB70:X=CU:Y=I:GOSUB20:NEXT
I
760 HF=0
770 RETURN
```

Play to foundations from open deck.

```
780 NU=OD(IN(7)-1):GOSUB70:FL=1
785 IF (F(SU)<>VA-1) AND (F(SU)=0) THEN GOSUB 1580:RETURN
790 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB1550:RETURN
800 GOSUB980
810 OD(IN(CU))=0
820 IF IN(CU)=0 THEN GOSUB 720:RETURN
830 GOSUB730
835 GOSUB150
840 RETURN
```

Play last card of column to foundation.

```
850 REM
860 X=CU:IF P(CU,0)<>0 THEN PRINT@X*8,G$(6);:GOTO 880
870 FOR Y=0 TO 2:PRINT@Y*64+X*8,STRING$(8,32);:NEXT Y
880 C(CU,0)=P(CU,0):IF P(CU,0)=0 THEN RETURN
890 NU=C(CU,0):X=CU:Y=0:GOSUB70:GOSUB20
900 IN(CU)=1
```

```
910 FORI=0TO4:P(CU,I)=P(CU,I+1):NEXTI:P(CU,5)=0
920 RETURN
```

Main subroutine to play to foundations.

```
930 IF HF<>0 THEN GOSUB 1480:RETURN
935 FL=0
940 IF IN(CU)=0 THEN GOSUB 1510:RETURN
950 IF CU=7 THEN GOSUB 780:RETURN
960 NU=C(CU,IN(CU)-1):GOSUB70
965 IF (F(SU)<>VA-1) AND (F(SU)=0) THEN GOSUB 1580:RETURN
970 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB1550:RETURN
980 X=7
990 Y=SU$3-3
1030 GOSUB20:F(SU)=VA
1040 IN(CU)=IN(CU)-1:IF FL<>0 THEN RETURN
1050 C(CU,IN(CU))=0
1060 IF IN(CU)=0 THEN GOSUB 850:RETURN
1070 X=CU:Y=IN(CU)-1:NU=C(CU,IN(CU)-1):GOSUB70:GOSUB20
1085 PRINT@CU$8+(IN(CU)+2)$64,STRING$(8,32);
1100 RETURN
```

Initialize variables.

```
1110 DIM C(6,11),P(6,5),D(51),OD(23),F(4),IN(7),TX$(13),G$(10)
1112 PRINT@342,CHR$(23);"SOLITAIRE"
1114 GOSUB 2000
1116 IF PEEK(293)=73 THEN Q1=94 ELSE Q1=91
1120 FORI=1TO13:READTX$(I):NEXTI
1150 FORI=0TO6:FORJ=0TO5:C(I,J)=0:P(I,J)=0:NEXTJ:FORJ=6TO11:C(I,
J)=0:NEXTJ,I
1160 FORI=0TO23:OD(I)=0:NEXTI
1170 FORI=0TO3:F(I)=0:NEXTI
1200 REM
1210 IN=0:FORI=1TO4:FORJ=1TO13:D(IN)=100$I+J:IN=IN+1:NEXTJ,I
```

Shuffle cards.

```
1230 FORI=51TO0STEP-1:X=INT(RND(0)$(I+1)):T=D(X):D(X)=D(I):D(I)=
T:NEXTI
```

Deal to tableau.

```
1240 IN=0:FORI=1TO6:FORJ=0TOI-1:P(I,J)=D(IN):IN=IN+1:NEXTJ,I
1250 FORI=0TO6:C(I,0)=D(IN):IN=IN+1:NEXTI
```

Draw set-up.

```
1260 CLS
1290 FORI=0TO6:NU=C(I,0):GOSUB70:X=I:Y=0:GOSUB20:NEXTI
1300 FORI=0TO6:IN(I)=1:NEXTI:IN(7)=0
1310 GOSUB100:GOSUB1460
1320 CU=0:DC=0:X=CU:GOSUB150
```

Control program.

```
1330 A$=INKEY$:IFA$=""THEN1330
1335 IF A$>=CHR$(96) AND A$<=CHR$(127) THEN A$=CHR$(ASC(A$)-32)
1340 IF A$=CHR$(9) THEN GOSUB200:GOTO1330
1350 IF A$=CHR$(8) THEN GOSUB240:GOTO1330
1360 IF A$="N" THEN GOSUB100:GOTO 1330
1370 IF A$="P" THEN GOSUB280:GOTO 1330
1380 IF A$="D" THEN GOSUB 410:GOTO 1330
1390 IF A$<>"F" THEN 1400 ELSE GOSUB 930: IF F(1)<13 OR F(2)<13
OR F(3)<13 OR F(4)<13 THEN 1330 ELSE PRINT@960,CHR$(30);"YOU WIN
!!!  CARE TO PLAY AGAIN? (Y/N)";
1395 PRINTCHR$(14);:A$=INKEY$:PRINTCHR$(15);:IFA$=""THEN1395ELSE
IFA$<>"N"ANDA$<>CHR$(110)THENRUNELSECLEAR50:CLS:END
1400 IF A$="E" THEN 1420
1410 GOTO 1330
```

Text messages.

```
1420 PRINT@960,CHR$(30);"DO YOU WANT TO END THE GAME? (Y/N) ";
1430 PRINTCHR$(14);:A$=INKEY$:PRINTCHR$(15);:IFA$=""THEN1430
1432 IF A$<>"Y" AND A$<>CHR$(121) THEN GOSUB 1460:GOTO 1330
1440 RUN
1459 STOP
1460 PRINT@960,CHR$(30);"ARROWS MOVE  D=DROP  N=NEXT  P=PICK UP
 E=END  F=FOUNDATION";
1470 RETURN
1480 A$="YOU ALREADY HAVE CARD(S) IN YOUR HAND.  'DROP' THEM.":G
OSUB2100:RETURN
1490 A$="THERE ARE NO MORE CARDS IN THE DECK.":GOSUB2100:RETURN
1500 A$="WHY DON'T YOU PLAY THE ACE ON A FOUNDATION!":GOSUB2100:
RETURN
1510 A$="THERE ARE NO CARDS HERE TO PICK UP.":GOSUB2100:RETURN
1520 A$="YOU DO NOT HAVE ANY CARDS TO DROP.":GOSUB2100:RETURN
1530 A$="YOU CAN'T PLAY A BLACK CARD ON A BLACK CARD.":GOSUB2100
:RETURN
1540 A$="YOU CAN'T PLAY A RED CARD ON A RED CARD.":GOSUB2100:RET
URN
```

```
1550 A$="YOU CAN'T PLAY A"+TX$(VA)+" ON A"+TX$(TV)+".":GOSUB2100
:RETURN
1560 A$="YOU CAN'T DROP THEM HERE.":GOSUB2100:RETURN
1570 A$="YOU CAN ONLY DROP A KING ON AN EMPTY SPACE.":GOSUB2100:
RETURN
1580 A$="YOU MUST START YOUR FOUNDATION WITH AN ACE.":GOSUB2100:
RETURN
```

Names of cards.

```
1590 DATA "N ACE"," TWO"," THREE"," FOUR"," FIVE"," SIX"," SEVEN
","N EIGHT"," NINE"," TEN"," JACK"," QUEEN"," KING"
1990 STOP
```

Initialization of graphics strings.

```
2000 G$=CHR$(26)+STRING$(6,8)
2010 G$(1)=CHR$(191)+CHR$(159)+CHR$(135)+CHR$(139)+CHR$(175)+CHR
$(191)+G$+CHR$(189)+CHR$(144)+CHR$(144)+CHR$(160)+CHR$(160)+CHR$
(190)+G$+CHR$(191)+CHR$(183)+CHR$(177)+CHR$(178)+CHR$(187)+CHR$(
191)
2020 G$(2)=CHR$(191)+CHR$(191)+CHR$(135)+CHR$(139)+CHR$(191)+CHR
$(191)+G$+CHR$(183)+CHR$(128)+CHR$(145)+CHR$(162)+CHR$(128)+CHR$
(187)+G$+CHR$(191)+CHR$(183)+CHR$(177)+CHR$(178)+CHR$(187)+CHR$(
191)
2030 G$(3)=CHR$(131)+CHR$(131)+CHR$(187)+CHR$(147)+CHR$(131)+CHR
$(171)+G$+" "+CHR$(174)+CHR$(191)+CHR$(191)+CHR$(132)+CHR$(170)+
G$+CHR$(176)+CHR$(176)+CHR$(187)+CHR$(177)+CHR$(176)+CHR$(186)
2040 G$(4)=CHR$(131)+CHR$(187)+CHR$(147)+CHR$(187)+CHR$(147)+CHR
$(171)+G$+CHR$(130)+CHR$(175)+CHR$(191)+CHR$(191)+CHR$(135)+CHR$
(170)+G$+CHR$(176)+CHR$(176)+CHR$(187)+CHR$(177)+CHR$(176)+CHR$(
186)
2050 G$(5)=CHR$(184)+CHR$(189)+CHR$(144)
2060 G$=G$+STRING$(2,8):G$(6)=STRING$(7,191)+CHR$(149)+G$+STRING
$(7,191)+CHR$(149)+G$+STRING$(7,191)+CHR$(149)
2070 VA$="A23456789TJQK"
2080 RETURN
```

General-purpose delay subroutine.

```
2100 PRINT@960,CHR$(30);A$;
2110 PRINT@1022,CHR$(143);:FORT=1TO5:IF INKEY$="" THEN NEXTT:PRI
NT@1022," ";:FOR T=1 TO 5:IF INKEY$="" THEN NEXT T:GOTO 2110
2120 GOSUB1460:RETURN
```

TRS-80® SWAT TABLE FOR:
SOLITAIRE

LINES	SWAT CODE	LENGTH
10 - 130	RV	265
140 - 270	KV	240
280 - 385	BD	283
390 - 490	SJ	303
500 - 600	AY	420
610 - 720	EB	274
730 - 830	DD	277
835 - 935	RC	263
940 - 1070	MA	313
1085 - 1210	XT	357
1230 - 1350	CC	380
1360 - 1459	DH	463
1460 - 1550	XC	557
1560 - 2020	PG	587
2030 - 2110	KG	519
2120 - 2120	YI	12

NOTES:

GAMBLER

by Randy Hawkins

Gambler is a game program for up to four players requiring a 16K TRS-80. An amplifier is optional to hear the accompanying sound.

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 TRS-80 will make all its own decisions, and will even touch its own keys as directed by the program. The only exception to this is the note at the bottom of the screen that asks you to "TOUCH ANY KEY TO CONTINUE". Usually, this delay is to allow you to read some instructions or to make a decision. Even when it is the computer's turn, it will wait for you to give the signal to continue.

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$: Graphics strings used as borders in game selection screen.
B$: User input via INKEY$.
D(6): Utility variables for six rolls of dice, six horse race positions, etc.
D$(6): Contains the pictures of the six dice.
GN: Game number selected in selection routine.
H$: Contains picture of the horses used in horse race subroutine.
HM: Contains highest money total when searching for winner.
HN: Number of player with highest money total.
I, J, K, L, S: Utility FOR-NEXT counters.
IO(X): Contains number of IOUs held by player X.
IP: Used as a loop variable to signal which player's turn it is.
JJ: Upper boundary on the FOR-NEXT loop that increases gradually to slow movement of the dot in game selection function.
L$(i,j): Lottery ticket string array.
M(I): Contains amount of money help by player I.
M$: Used to hold the message printed out during various functions.
N: Contains the number of players.
N$(i): Contains players' names.
PB: Contains the number of IOUs which can presently be paid off.
Q: Dummy variable used with the USR(0)function to create sound.
QQ: Used in lines 890 and 900 to create the four-not tune played as message M$ is displayed.
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,: 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        TRS-80 BASIC       SS
SS          'Gambler'        SS
SS   Author:  Randy Hawkins  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$, graphics strings used in the borders of the game-selection screen. Subroutine 420 does further initialization of graphics. Lines 30 and 40 set the number of players, N. Line 50 determines players' names and initializes money, M(I), number of IOUs, IO(I), and lottery tickets, L$. If desired, instructions are displayed after prompting in line 50.

```
10 CLS:PRINTCHR$(23)"WELCOME TO *GAMBLER*":PRINT"(WITH SOUND EFF
ECTS)":PRINT
20 CLEAR1000:DEFINTA-Z:A1$=CHR$(191)+"  "+CHR$(149)+STRING$(17,3
```

```
2):A1$=A1$+A1$+A1$+CHR$(191):A2$=CHR$(188)+STRING$(2,140)+CHR$(1
56)+STRING$(17,140):A2$=A2$+A2$+A2$+CHR$(188):GOSUB420:GOSUB1390
30 PRINT"HOW MANY PLAYERS (UP TO 4) ? ";
40 B$=INKEY$:IFB$=""THEN40ELSEN=VAL(B$):IFN<1ORN>4THEN40ELSEPRIN
TN
50 FORI=1TON:PRINT"WHO IS PLAYER #";I;:INPUTN$(I):M(I)=100:IO(I)
=0:L$(I,0)="":NEXTI:N$(I)="TRS-80":M(I)=100:IO(I)=0:N=N+1:PRINT:
PRINT"DO YOU NEED TO SEE INSTRUCTIONS?"
60 B$=INKEY$:IFB$=""THEN60ELSEIF(ASC(B$)AND95)=89THENGOSUB1370
```

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 FORIP=1TON
80 CLS:PRINTCHR$(23);"PRESENT BANKROLLS            IOUS";STRING$(
32,45);:FORJ=1TON:PRINTUSINGN$(J)+" "+STRING$(15-LEN(N$(J)),46)+
"$$###";M(J);:PRINT@122+J*64,IO(J);:NEXTJ:GOSUB200:PRINT:PRINT"I
T IS ";N$(IP);"'S TURN.":IFM(IP)>=0THEN100
90 IO(IP)=IO(IP)+1:M(IP)=M(IP)+100:IFM(IP)<0THEN90ELSEPRINT"YOU
MUST BORROW MONEY TO CONT-":PRINT"INUE TO PLAY AND PAY IT BACK T
O":PRINT"THE BANK ... PLUS INTEREST!":FORTI=1TO2000:NEXTTI:GOTO8
0
100 IFIO(IP)>0ANDM(IP)>=110THENPB=INT(M(IP)/110)ELSE130
110 IFPB>IO(IP)THENPB=IO(IP)
120 IO(IP)=IO(IP)-PB:M(IP)=M(IP)-110*PB:PRINT"YOU CAN PAY BACK";
PB;"IOU NOTE":PRINT"AT 110 DOLLARS EACH.":FORTI=1TO2000:NEXTTI:G
OTO80
130 IFL$(IP,0)="XXX"THEN160ELSEPRINT"WOULD YOU LIKE TO BUY A LOT
TERY":PRINT"TICKET FOR $10 ? ";
140 IFIP=NTHENPRINT"Y":GOSUB260:L$(IP,0)=L$(IP,0)+"X":L$(IP,LEN(
L$(IP,0)))=T$:M(IP)=M(IP)-10:FORTI=1TO1000:NEXTTI:GOTO80
150 B$=INKEY$:IFB$=""THEN150ELSEB$=CHR$(ASC(B$)AND95):IFB$="N"TH
ENPRINT"N":GOTO160ELSEIFB$<>"Y"THEN150ELSEPRINT"Y":GOSUB260:L$(I
P,0)=L$(IP,0)+"X":L$(IP,LEN(L$(IP,0)))=T$:M(IP)=M(IP)-10:FORTI=1
TO1000:NEXTTI:GOTO80
160 PRINT@962,"TOUCH ANY KEY TO CONTINUE";
170 B$=INKEY$:IFB$=""THEN170ELSEGOSUB290
180 ONGNGOSUB1020,810,460,1130,1170,1190,460,1200,1230,1240,460,
750,1300,940,1020,710,810,1310,1320,750,460,1330,710,810,1340,13
50,1330,810,460,940,1310,710,750,1360,810,710
```

```
190 NEXTIP:GOTO70
```

Subroutine 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:FORJ=2TON:IFM(J)>HMTHENHM=M(J):HN=J:NEXTJELSENE
XTJ
210 IFHM<1000THENRETURN
220 K=0:FORJ=1TON:IFM(J)=HMTHENK=K+1:NEXTJELSENEXTJ
230 IFK>1THENPRINT"SINCE THERE'S A TIE, KEEP GOING";:RETURN
240 PRINT:PRINT" ";N$(HN);" WINS THIS GAME WITH":PRINTHM;"DOLLAR
S!":FORI=0TO90STEP2:Q=USR(11141-I):NEXTI:PRINTCHR$(23):PRINT"WOU
LD YOU LIKE TO PLAY AGAIN ?"
250 B$=INKEY$:IFB$=""THEN250ELSEB$=CHR$(ASC(B$)AND95):IFB$="Y"TH
ENRUNELSEEND
```

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$=CHR$(48+RND(5)):T1$=CHR$(48+RND(6)):IFT1$<=T$THEN260ELSET
$=T$+"-"+T1$:FORI=1TON:IFL$(I,0)=""THEN280
270 FORJ=1TOLEN(L$(I,0)):IFT$=L$(I,J)THEN260ELSENEXTJ
280 NEXTI:PRINT"YOU RECEIVE TICKET NUMBER ";T$:FORJ=1TO20:OUT255
,7:OUT255,8:NEXTJ:RETURN
```

Game selector. Prints border, using A1$ and A2$. Displays names of games read 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:CLS:PRINTA2$;:FORJ=1TO12:PRINTA1$;:NEXTJ:FORI=1TO3:P
RINTCHR$(143)STRING$(2,140)CHR$(141)STRING$(17,140);:NEXT:PRINTC
HR$(143)
300 FORX=68TO772STEP64:READA$:PRINT@X,A$;:READA$:PRINT@X+21,A$;:
READA$:PRINT@X+42,A$;:NEXTX
310 X=65:B$=INKEY$:PRINT@970,N$(IP);", TOUCH ANY KEY FOR YOUR SE
LECTION";
320 Y=X:PRINT@X,STRING$(2,143);:Q=USR(5000+RND(128)):X=RND(12)*6
4+RND(3)*21-20
330 IFIP=NTHENIFRND(30)=25THEN360ELSE350
340 B$=INKEY$:IFB$<>""THEN360
350 PRINT@Y,"  ";:GOTO320
```

```
360 JJ=100:KK=5160:FORI=1TO5+RND(4):Q=USR(KK):KK=KK+20:FORJ=1TOJ
J:NEXTJ:JJ=JJ+100:PRINT@Y,"  ";:Y=Y:PRINT@X,STRING$(2,143);:X=X+
21:IFPOS(0)>40THENX=X+1
370 IFX>830THENX=65:NEXTIELSENEXTI
380 Q=USR(KK):FORTI=1TO300:NEXTTI:GN=INT(Y/21)-2:PRINT@Y,"  ";:F
ORI=1TO5:Q=USR(0):PRINT@Y,STRING$(2,143);:FORTI=1TO99:NEXTTI:PRI
NT@Y,"  ";:NEXTI:RETURN
390 DATAPOKER PARTY,F O R T U N E !,SWEEPSTAKES!,LOVE THY NEIGHB
OR,EASY COME EASY GO,WIN A FEW,SWEEPSTAKES!,UNLUCKY SEVEN,LOSE A
 FEW
400 DATAEVEN STEVEN,SWEEPSTAKES!,DAILY DOUBLE,POT LUCK,HIGH ROLL
ER,POKER PARTY,LOTTERY,F O R T U N E !,JACKPOT,$100 BONUS,OFF TO
 THE RACES,SWEEPSTAKES!,LOSE THIS TURN
410 DATALOTTERY,F O R T U N E !,TAX TIME,BONANZA,LOSE THIS TURN,
F O R T U N E !,SWEEPSTAKES!,HIGH ROLLER,JACKPOT,LOTTERY,HORSE R
ACE,MAD MONEY,F O R T U N E !,LOTTERY
```

Graphics initialization. Set up graphics array D$, which consists of pictures of the six dice.

```
420 R$=CHR$(26)+STRING$(7,24):D$(1)=STRING$(3,191)+CHR$(143)+STR
ING$(3,191)+R$+STRING$(7,143):D$(2)=CHR$(191)+CHR$(179)+STRING$(
5,191)+R$+STRING$(5,143)+CHR$(140)+CHR$(143)
430 D$(3)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(143)+STRING$(3,191)
+R$+STRING$(5,143)+CHR$(140)+CHR$(143):D$(4)=CHR$(191)+CHR$(179)
+STRING$(3,191)+CHR$(179)+CHR$(191)+R$+CHR$(143)+CHR$(140)+STRIN
G$(3,143)+CHR$(140)+CHR$(143)
440 D$(5)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(143)+CHR$(191)+CHR$
(179)+CHR$(191)+R$+CHR$(143)+CHR$(140)+STRING$(3,143)+CHR$(140)+
CHR$(143):D$(6)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(179)+CHR$(191
)+CHR$(179)+CHR$(191)+R$+CHR$(143)+CHR$(140)+CHR$(143)
450 D$(6)=D$(6)+CHR$(140)+CHR$(143)+CHR$(140)+CHR$(143):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). 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. Results are displayed in line 690.

```
460 CLS:PRINT@19,"S W E E P S T A K E S !":PRINT@70,CHR$(176)STR
ING$(8,140)" G "STRING$(8,140)CHR$(176)STRING$(8,140)" H "STRING
$(8,140)CHR$(176):PRINT@134,"A":PRINT@154,"C":PRINT@174,"E":PRIN
T@198,CHR$(143)STRING$(19,32)CHR$(143)STRING$(19,32)CHR$(143)
```

```
470 PRINT@400,CHR$(191)STRING$(19,32)CHR$(191)STRING$(19,32)CHR$
(191):PRINT@464,"B":PRINT@484,"D":PRINT@504,"F":PRINT@528,CHR$(1
31)STRING$(8,140)" I "STRING$(8,140)CHR$(131)STRING$(8,140)" J "
STRING$(8,140)CHR$(131):PRINT@448,"P = PAIR";
480 PRINT@512,"S = STRAIGHT":FORX=1TO6:PRINT@249+X*10,D$(X);:NEX
TX
490 PRINT@650,"DO YOU NEED SWEEPSTAKES INSTRUCTIONS ?":B$=INKEY$
500 B$=INKEY$:IFB$<>""THEN510ELSEIFRND(9)=1THEN0=USR(1E4+RND(99)
):GOTO500ELSE500
510 B$=CHR$(ASC(B$)AND95):IFB$="N"THEN540ELSEPRINT@640,CHR$(31);
"IN SWEEPSTAKES EVERYONE ANTES $10.  EACH PLAYER BETS ON THE OUT
-COME OF THE ROLL OF 6 DICE.  THERE ARE THREE TYPES OF BETS.":GO
SUB1470
520 PRINT@640,CHR$(31);"ONE WAY IS TO BET ON WHERE THE HIGHEST N
UMBER ROLLED WILL APPEARIF THE HIGH NUMBER APPEARS IN A,B,C,D,E,
OR F YOU WIN $300.  LET-TERS G,H,I,AND J COVER TWO SPOTS.  IF YO
U BET G, YOU COVER SPOTSA AND C AND CAN WIN $150."
530 GOSUB1470:PRINT@640,CHR$(31);"ANOTHER BET IS <<P>> FOR PAIRS
.  IF TWO DICE NEXT TO ONE ANOTHERMATCH, YOU WIN $200.  THE THIR
D BET IS <<S>> FOR A STRAIGHT.    IF 3 CONSECUTIVE DICE APPEAR I
N NUMERICAL ORDER, YOU WIN $450.":GOSUB1470
540 PRINT@640,CHR$(31):PRINT@690,"A-F $300":PRINT@754,"G-J $150"
:PRINT@820,"P $200":PRINT@884,"S $450":FORI=1TON:M(I)=M(I)-10:S$
(I)="":NEXTI:X=640:PRINT@980,"PLACE YOUR BETS:";
550 FORJ=1PTON:PRINT@X,N$(J);" - ";:X=X+64
560 IFJ=NTHENB$=CHR$(64+RND(11)):IFB$="K"THENB$="P":GOTO580ELSE5
80
570 B$=INKEY$:IFB$=""THEN570ELSEB$=CHR$(ASC(B$)AND95):IF(B$>"@"A
NDB$<"K")ORB$="P"ORB$="S"THEN580ELSE570
580 FORK=1TON:IFB$=S$(K)THEN560ELSENEXTK:PRINTB$;:S$(J)=B$:NEXTJ
590 IFIP=1THENHN=1:GOTO620ELSEFORJ=1TOIP-1:PRINT@X,N$(J);" - ";:
X=X+64
600 B$=INKEY$:IFB$=""THEN600ELSEB$=CHR$(ASC(B$)AND95):IF(B$>"@"A
NDB$<"K")ORB$="P"ORB$="S"THEN610ELSE600
610 FORK=1TON:IFB$=S$(K)THEN600ELSENEXTK:PRINTB$;:S$(J)=B$:NEXTJ
:HN=1
620 GOSUB1470:PRINT@192,:PRINT@256,:FORX=1TO6:FORL=1TO5:D=USR(5E
3+RND(128)):D(X)=RND(6):PRINT@249+X*10,D$(D(X));:NEXTL:IFD(X)>HN
THENHN=D(X):NEXTXELSENEXTX
630 FORJ=1TON:IFS$(J)="P"THENS$(J)="SORRY, NO PAIRS!":FORK=1TO5:
IFD(K)=D(K+1)THENS$(J)=" $200 FOR A PAIR":M(J)=M(J)+200:K=5:NEXT
KELSENEXTK
640 IFS$(J)="S"THENFORK=1TO4:IFD(K)=D(K+1)-1ANDD(K)=D(K+2)-2THEN
S$(J)=" $450 FOR THE STRAIGHT":M(J)=M(J)+450:K=4:NEXTKELSENEXTK
650 IFS$(J)="S"THENS$(J)="NO STRAIGHT!":FORK=1TO4:IFD(K)=D(K+1)+
```

```
1ANDD(K)=D(K+2)+2THENS$(J)=" $450 FOR THE STRAIGHT":M(J)=M(J)+45
0:D=4:NEXTKELSENEXTK
660 IF(S$(J)="G"AND(D(1)=HNORD(3)=HN))OR(S$(J)="H"AND(D(3)=HNORD
(5)=HN))OR(S$(J)="I"AND(D(2)=HNORD(4)=HN))OR(S$(J)="J"AND(D(4)=H
NORD(6)=HN))THENS$(J)=" $150 FOR BET ON "+S$(J):M(J)=M(J)+150
670 IFS$(J)>"@"ANDS$(J)<"G"THENK=ASC(S$(J))-64:IFD(K)=HNTHENS$(J
)=" $300 ON SPOT "+S$(J):M(J)=M(J)+300ELSES$(J)=" YOU LOSE WITH
SPOT "+S$(J)
680 IFLEN(S$(J))=1THENS$(J)=" MAYBE NEXT TIME!"
690 NEXTJ:PRINT@576,CHR$(30):X=640:FORJ=1PTON:PRINT@X,N$(J);" -
";S$(J);CHR$(31):X=X+64:NEXTJ:IFIP=1THEN700ELSEFORJ=1TOIP-1:PRIN
T@X,N$(J);" - ";S$(J);:X=X+64:NEXTJ
700 GOSUB1470:RETURN
```

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

```
710 CLS:PRINT@24,"L O T T E R Y":PRINT:PRINT:PRINT"IF BOTH THE N
UMBERS ON YOUR TICKET ARE AMONG THE SIX NUMBERS"
715 PRINT"ROLLED, YOU WILL RECEIVE $50 FOR THAT TICKET.  HERE WE
 GO ...":FORJ=1TON:PRINT@576+J*64,N$(J)TAB(20)L$(J,1)TAB(30)L$(J
,2)TAB(40)L$(J,3):NEXTJ:GOSUB1470
720 FORJ=1TO6:D(J)=0:NEXTJ:FORJ=1TO6:K=RND(6):PRINT@313+J*10,D$(
K);:D(K)=1:Q=USR(1000*(J+1)):NEXTJ
730 FORJ=1TON:FORK=1TOLEN(L$(J,0)):FORL=1TO300:NEXTL:IFD(VAL(LEF
T$(L$(J,K),1)))=1ANDD(VAL(RIGHT$(L$(J,K),1)))=1THENM(J)=M(J)+50:
Q=USR(0):PRINT@586+J*64+K*10,"$50";:NEXTK,JELSEFORL=2032TO2080:Q
=USR(L):NEXTL:PRINT@586+J*64+K*10,"---";:NEXTK,J
740 GOSUB1470:FORJ=1TON:L$(J,0)="":FORK=1TO3:L$(J,K)="":NEXTK,J:
RETURN
```

Horse race routine. The picture of a horse is created 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 POS(0) passes 61 in line 780, the winner is announced in lines 790 and 800.

```
750 CLS:H$=CHR$(32)+"-"+CHR$(156)+CHR$(140)+CHR$(150)+CHR$(129):
FORJ=1TO6:D(J)=10+J*128:PRINT@J*128,J;TAB(10);H$;:NEXTJ:FORJ=1TO
N:PRINT@J*128,N$(J);:M(J)=M(J)-20:NEXTJ:PRINT@19,"H O R S E   R
A C E S !";:FORY=7TO37STEP3:SET(120,Y):NEXTY
760 PRINT@900,"EVERYONE HAS BET $20, THE WINNER WILL RECEIVE $10
0.":GOSUB1470
```

```
770 FORJ=1TO3:Q=USR(266):FORK=1TO20:NEXTK,J
780 FORI=1TO3:K=RND(6):D(K)=D(K)+1:PRINT@D(K),H$;:IFPOS(0)>61THE
NI=3:NEXTELSENEXT:GOTO770
790 PRINT@896,CHR$(31);:IFK<=NTHENPRINTTAB(23-LEN(N$(K))/2);N$(K
);" ";:M(K)=M(K)+100ELSEPRINTTAB(16);"HORSE NUMBER";K;
800 PRINT"WINS THIS RACE!":GOSUB1470: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$="":CLS:PRINT@262,"T H E  F O R T U N E  T E L L E R  S
 A Y S :":K=1:M=10*RND(5)+50
820 IFRND(5)=5THENM$="HOLD A SWEEPSTAKES.":K=2:GOTO890
830 IFRND(5)=5THENM$="HOLD A LOTTERY.":K=3:GOTO890
840 IFRND(2)=2THENM$="COLLECT FROM ":K=-1ELSEM$="PAY TO "
850 J=RND(N+1):IFJ<>IPTHEN860ELSEM$=M$+"EVERYONE":M(IP)=M(IP)-K*
N*M:FORJ=1TON:M(J)=M(J)+K*M:NEXTJ:GOTO880
860 IFJ=N+1THENM$=M$+"THE BANK":M(IP)=M(IP)-K*M:GOTO880
870 M$=M$+N$(J):M(IP)=M(IP)-K*M:M(J)=M(J)+K*M
880 M$=M$+STR$(M)+" DOLLARS."
890 QQ=0:D(0)=5000:D(1)=5040:D(2)=5020:D(3)=5007:PRINTCHR$(23):X
=USR(266)
900 M$=STRING$(32,32)+N$(IP)+", "+M$+STRING$(32,32):FORJ=1TOLEN(
M$)-31:PRINT@512,MID$(M$,J,31);:Q=USR(D(QQAND3)):QQ=QQ+1:FORTI=1
TO55:NEXTTI,J:M$=""
910 PRINT@966,"TOUCH ANY KEY TO CONTINUE";:B$=INKEY$
920 IFINKEY$=""THEN920
930 IFK=2THEN460ELSEIFK=3THEN710ELSERETURN
```

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 CLS:PRINT" H I G H  R O L L E R -- EVERYONE ANTES $20 AND R
OLLS TWO DICE. THE HIGHEST TOTAL TAKES THE $100 PRIZE.  TO ROLL
DICE, TOUCH":PRINT" ANY KEY ON YOUR TURN.":GOSUB1470:PRINT@192,C
HR$(31)
950 FORJ=1TON:M(J)=M(J)-20:X=64+J*128:PRINT@X,N$(J):X=X+10:GOSUB
1450:D(J)=K:X=X+15:GOSUB1450:D(J)=D(J)+K:X=X-25:PRINT@X+35,"YOUR
 TOTAL IS";D(J);:NEXTJ
960 HN=D(1):HM=1:FORJ=2TON:IFD(J)>HNTHENHN=D(J):HM=J:NEXTJELSENE
XTJ
970 K=0:FORJ=1TON:IFD(J)=HNTHENK=K+1:NEXTJELSENEXTJ
980 IFK=1THENPRINT@896,TAB(19+LEN(N$(HM))/2);N$(HM);" WINS THE P
```

```
OT!":M(HM)=M(HM)+100:B$=INKEY$:GOSUB1470:RETURN
990 PRINT@900,"WE'VE GOT A TIE!  THOSE HIGH ROLLERS WILL ROLL AG
AIN!":B$=INKEY$:GOSUB1470:PRINT@192,CHR$(31);:X=192
1000 FORJ=1TON:IFD(J)<HNTHEND(J)=0:NEXTJELSEPRINT@X,N$(J):X=X+10
:GOSUB1450:D(J)=K:X=X+15:GOSUB1450:D(J)=D(J)+K:X=X-25:PRINT@X+35
,"YOUR TOTAL IS";D(J);:X=X+128:NEXTJ
1010 GOTO960
```

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 CLS:PRINT"P O K E R   P A R T Y --  EACH PLAYER PAYS $20 AN
D ROLLS THREE  DICE.  THE BEST POKER HAND (THREE OF A KIND > STR
AIGHT > PAIR)  WINS $100.   TOUCH ANY KEY TO ROLL DICE.":GOSUB14
70:PRINT@192,CHR$(31):XX=256:FORI=1TON:M(I)=M(I)-20:NEXTI:HM=0
1030 FORI=1TON:PRINT@XX,N$(I):FORM=1TO3:J=I:X=XX+10$M:B$=INKEY$:
GOSUB1450:D(M)=K:NEXTM:P(I)=0
1040 IFD(1)=D(2)ANDD(1)=D(3)THENPRINT@XX+40,"THREE OF A KIND!";:
P(I)=30+D(1):GOTO1100
1050 IFD(1)=D(2)THENP(I)=10+D(1)ELSEIFD(1)=D(3)THENP(I)=10+D(1)E
LSEIFD(2)=D(3)THENP(I)=10+D(2)
1060 IFP(I)>0THENPRINT@XX+40,"A PAIR!";:GOTO1100
1070 FORJ=1TO3:IF(D(1)=D(2)+1ANDD(1)=D(3)+2)OR(D(1)=D(2)-1ANDD(1
)=D(3)-2)THENP(I)=20:NEXTJELSEHN=D(1):D(1)=D(2):D(2)=D(3):D(3)=H
N:NEXTJ
1080 HN=D(1):FORJ=2TO3:IFD(J)>HNTHENHN=D(J):NEXTJELSENEXTJ
1090 P(I)=P(I)+HN:IFP(I)>20THENPRINT@XX+40,"A STRAIGHT!";ELSEPRI
NT@XX+40,"HIGHEST ROLL IS A";P(I);
1100 XX=XX+128:IFP(I)>HMTHENHM=P(I):NEXTIELSENEXTI
1110 K=0:FORI=1TON:IFP(I)=HMTHENK=K+1:J=I:NEXTIELSENEXTI
1120 IFK=1THENPRINT@914,N$(J);" WINS THE POT OF $100":M(J)=M(J)+
100:GOSUB1470:RETURNELSEPRINT@905,"WE HAVE A TIE ... SO LET'S AL
L PLAY ANOTHER HAND!":GOSUB1470:PRINT@196,CHR$(31):XX=256:HM=0:G
OTO1030
```

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

```
1130 CLS:PRINT@256,CHR$(23):K=0:M=RND(5)*10+50
1140 J=RND(N):IFJ=IPTHEN1140
1150 M$="SHOW THAT YOU'RE A GOOD NEIGHBOR AND GIVE "+N$(J)+STR$(
M)+" DOLLARS."
1160 M(IP)=M(IP)-M:M(J)=M(J)+M:GOTO890
```

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

```
1170 CLS:PRINT"E A S Y   C O M E   E A S Y   G O  --  THE BANK W
ILL PAY YOU 10 TIMES ROLL OF 2 DICE.  TOUCH ANY KEY TO ROLL DICE
, ";N$(IP);".":GOSUB1470:J=IP:X=279:GOSUB1450:M=K:X=289:GOSUB145
0:M=(M+K)*10:M(IP)=M(IP)+M:PRINT@470,"YOU WIN";M;"DOLLARS."
1180 GOSUB1470:RETURN
```

Short-message 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 CLS:PRINT"W I N   A   F E W  -- THE BANK WILL PAY YOU TEN T
IMES":PRINT"THE ROLL OF ONE DIE.  TOUCH ANY KEY TO ROLL DIE, ";N
$(IP);".":GOSUB1470:J=IP:X=284:GOSUB1450:M=K*10:M(IP)=M(IP)+M:PR
INT@470,"YOU WIN";M;"DOLLARS.":GOSUB1470:RETURN
1200 CLS:PRINT"U N L U C K Y   S E V E N  --  ROLL TWO DICE. IF
THE TOTAL IS":PRINT"SEVEN YOU LOSE $100.  FOR ANY OTHER TOTAL, Y
OU WIN $100.":PRINT"TOUCH ANY KEY TO ROLL DICE, ";N$(IP);".":GOS
UB1470:J=IP:X=279:GOSUB1450:M=K:X=289:GOSUB1450:M=M+K
1210 IFM=7THENPRINT@470,"YOU LOSE 100 DOLLARS!":M(IP)=M(IP)-100E
LSEPRINT@470,"YOU WIN 100 DOLLARS!":M(IP)=M(IP)+100
1220 GOSUB1470:RETURN
1230 CLS:PRINT"L O S E   A   F E W  --  YOU MUST PAY THE BANK TE
N TIMES THE   ROLL OF ONE DIE.  TOUCH ANY KEY TO ROLL DIE, ";N$
(IP);".":GOSUB1470:J=IP:X=284:GOSUB1450:M=K*10:M(IP)=M(IP)-M:PRI
NT@470,"YOU LOST";M;"DOLLARS.":GOSUB1470:RETURN
1240 CLS:PRINT"E V E N   S T E V E N  --  YOU MAY BET UP TO $90
AND ROLL TWO   DICE.  IF THE TOTAL IS EVEN, YOU COLLECT TWICE YO
UR BET.  TOUCH ANY KEY TO ROLL DICE.":PRINTN$(IP);", HOW MUCH WI
LL YOU BET ? ";:IFIP=NTHENB$=STR$(RND(4)+5):GOTO1260
1250 B$=INKEY$:IFB$=""ORVAL(B$)<1THEN1250
1260 M=VAL(B$)*10:PRINTM;"DOLLARS":GOSUB1470
1270 X=407:J=IP:GOSUB1450:P=K:X=417:GOSUB1450:P=P+K
1280 IFINT(P/2)*2=PTHENPRINT@598,"YOU WIN";2*M;"DOLLARS.":M(IP)=
M(IP)+2*MELSEPRINT@598,"YOU LOSE";M;"DOLLARS.":M(IP)=M(IP)-M
1290 GOSUB1470:RETURN
1300 CLS:PRINTCHR$(23):M=10*RND(5)+10:M(IP)=M(IP)-M:M$="WHY DON'
T YOU SWEETEN THE POT BY GIVING THE BANK"+STR$(M)+" DOLLARS.":K=
0:GOTO890
1310 CLS:PRINTCHR$(23):M=RND(5)*10+50:M$="YOU HIT THE JACKPOT! C
OLLECT"+STR$(M)+" DOLLARS.":K=0:M(IP)=M(IP)+M:GOTO890
```

```
1320 CLS:PRINTCHR$(23):M(IP)=M(IP)+100:M$="PLEASE ACCEPT THIS $1
00 BONUS!":K=0:GOTO890
1330 CLS:PRINTCHR$(23):M$="YOU JUST LOST THIS TURN!":K=0:GOTO890
1340 CLS:PRINTCHR$(23):M$="PAY $100 TAX TO THE BANK!":M(IP)=M(IP
)-100:K=0:GOTO890
1350 CLS:PRINTCHR$(23):M=RND(5)*10:M(IP)=M(IP)+M*N:FORJ=1TON:M(J
)=M(J)-M:NEXTJ:M$="WHAT A BONANZA! EVERYONE PAYS YOU"+STR$(M)+"
DOLLARS.":K=0:GOTO890
1360 CLS:PRINTCHR$(23):M=RND(5)*10:M(IP)=M(IP)-M*N:FORJ=1TON:M(J
)=M(J)+M:NEXTJ:M$="THIS OUGHT TO MAKE YOU MAD.  GIVE EVERYBODY"+
STR$(M)+" DOLLARS.":K=0:GOTO890
```

Instructions.

```
1370 CLS:PRINTCHR$(23)"THIS IS THE GAME OF GAMBLER!":PRINT:PRINT
"YOU AND THE TRS-80 ARE IN A CON-TEST TO SEE WHO WILL BUILD HIS
 $100 BANKROLL INTO $1000 FIRST. MONEY IS MADE AND LOST THROUGH
ASERIES OF GAMES OF CHANCE --"
1375 PRINT"FROM HORSE RACING AND DICE GAMES";
1380 PRINT"TO LOTTERIES AND SWEEPSTAKES! IFYOU SHOULD LOSE ALL Y
OUR MONEY":PRINT"YOUR IOU WILL BE ACCEPTED ( AS':PRINT"LONG AS Y
OU PAY IT BACK WITH":PRINT"INTEREST ). GOOD LUCK!":K=0:GOTO910
```

Utility subroutines. Lines 1390-1440 set up the sound routine in SOUND$. Lines 1450-1460 wait for input before stopping on one of the six dice at print position X. The computer's dice choice is made in line 1450. Lines 1470 and 1480 print a message and wait for keyboard response.

```
1390 SOUND$="////////////////////////////":REM 28 SLASHES
1400 I=VARPTR(SOUND$):J!=PEEK(I+1)+256*PEEK(I+2):J=J!+65535*(J!)
32767)
1410 FORK=1TO36:READA$:NEXTK:FORK=JTOJ+26:READX:POKEK,X:NEXTK
1420 IFPEEK(16396)=201THENPOKE16526,PEEK(I+1):POKE16527,PEEK(I+2
)ELSECMD"T":DEFUSR0=J:POKE14308,0
1430 RETURN
1440 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60,105,211
,255,45,32,253,13,16,238,175,211,255,201
1450 B$=INKEY$:IFJ=NTHENFORL=1TORND(50):K=RND(6):PRINT@X,D$(K);:
Q=USR(1E4+RND(99)):NEXTL:RETURN
1460 K=RND(6):PRINT@X,D$(K);:Q=USR(1E4+RND(99)):IFINKEY$=""THEN1
460ELSERETURN
1470 PRINT@978,"TOUCH ANY KEY TO CONTINUE";
1480 IFINKEY$<>""THENPRINT@978,CHR$(30);:RETURNELSEIFRND(9)=1THE
NQ=USR(1E4+RND(99)):GOTO1480ELSE1480
```

TRS-80® SWAT TABLE FOR:
GAMBLER

LINES	SWAT CODE	LENGTH
10 - 80	EV	663
90 - 140	TA	536
150 - 220	ZE	508
230 - 300	DD	552
310 - 390	TW	572
400 - 430	FI	602
440 - 470	UC	558
480 - 520	WK	576
530 - 570	PR	526
580 - 640	DC	564
650 - 690	CV	593
700 - 730	QO	503
740 - 800	WV	523
810 - 900	PH	613
910 - 980	XU	567
990 - 1030	MR	540
1040 - 1110	KT	508
1120 - 1170	VN	558
1180 - 1210	ER	540
1220 - 1260	SD	509
1270 - 1330	LK	509
1340 - 1370	NM	536
1375 - 1440	JO	540
1450 - 1480	ML	215

NOTES:

Operation: Sabotage

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

***Operation: Sabotage* is a fantasy/adventure game for an Apple with 16K 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 fantasy/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.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        TRS-80 BASIC         SS
SS   'Operation:  Sabotage'    SS
SS      Author: Ray Sato       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.

```
2 GOTO2610
```

Decode and print output.

```
4 IFP$=""THENRETURNELSEFORP=1TOLEN(P$):II=ASC(MID$(P$,P,N1)):PRI
NTCHR$(ABS((C155$(II>C64))-II));:NEXT:PRINT:RETURN
```

Encode input.

```
6 V$="":IFV0$=""THENRETURNELSEFORJ=1TOLEN(V0$):II=ASC(MID$(V0$,J
,N1)):V$=V$+CHR$(ABS((C155$(II>64))-II)):NEXT:RETURN
```

Display message, then end current turn.

```
8 GOSUB4:GOTO2250
```

Descriptions of individual rooms.

```
10 A$="ZM ZRIOLXP. GSVIV RH Z YOFV YFGGLM SVIV":S=2:RETURN
20 A$="Z MZIILD XLIIRWLI":N=1:S=3:RETURN
30 A$="Z MZIILD XLIIRWLI":N=2:S=4:RETURN
```

```
40 A$="Z MZIILD XLIIRWLI":N=3:S=5:RETURN
50 A$="Z HNZOO ILLN":N=4:S=6:RETURN
60 A$="Z WVXLMGZNRMZGRLM XSZNYVI":B$="GSVIV RH Z YOFV YFGGLM SVI
V":N=5:S=7:RETURN
70 A$="Z HNZOO HGLIZTV XSZNYVI":N=6:S=8:W=12:RETURN
80 A$="Z HNZOO XSZNYVI":N=7:S=9:W=13:RETURN
90 A$="Z HNZOO VOVXGILMRX OZYLIZGLIB":N=8:S=10:W=14:RETURN
100 A$="Z YRLOLTRXZO OZYLIZGLIB. GSVIV RH Z IVW YFGGLM LM GSV DZ
OO":N=9:W=15:RETURN
110 A$="Z HGLIZTV XSZNYVI":W=16:RETURN
120 A$="Z OZITV XSZNYVI. GSVIV RH Z XZYRMVG SVIV":S=13:W=17:E=7:
RETURN
130 A$="Z HGIZMTV KFIKOV ILLN. GSVIV RH Z YOFV YFGGLM SVIV":N=12
:S=14:W=18:E=8:RETURN
140 A$="Z HNZOO LUURXV":N=13:S=15:E=9:RETURN
150 A$="Z HNZOO ILLN DRGS Z XZIW GZYOV RM GSV XVMGVI":N=14:S=16:
W=20:E=10:RETURN
160 A$="Z OZITV LUURXV. GSVIV RH Z WVHP SVIV":N=15:W=21:E=11:RET
URN
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 HXIVVM IVHGH LM GSV DZOO":B$="G
SVIV RH Z YOFV ZMW Z IVW YFGGLM FMWVI GSV HXIVVM":N=19:S=21:E=15
:RETURN
210 A$="Z WZGZ IVXLIW HGLIZTV ILLN":N=20:E=16:RETURN
220 A$="IZWZI XLMGILO. GSVIV RH Z HNZOO HXIVVM SVIV":S=23:W=27:R
ETURN
230 A$="Z NVWRXZO HGZGRLM. GSVIV RH Z OZITV GZYOV SVIV":N=22:S=2
4:W=28:RETURN
240 A$="Z HVXFIRGB HGZGRLM":N=23:RETURN
250 A$="Z IZWRL ILLN":S=26:W=30:RETURN
260 A$="Z HNZOO ILLN. GSVIV RH Z HZUV RM GSV HLFGS DZOO":N=25:W=
31:RETURN
270 A$="GSV ILYLG XLMGILO XVMGVI":B$="GSVIV RH Z HNZOO XLMGILO X
LNKFGVI NLFMGVW RM GSV DZOO":W=32:E=22:RETURN
280 A$="GSV DVZKLMH HGLIZTV ILLN":S=29:W=33:E=23:RETURN
290 A$="Z ORYIZIB":N=28:S=30:W=34:RETURN
300 A$="Z HVXFIRGB XSVXP ZIVZ":N=29:W=36:E=25:RETURN
310 A$="Z HNZOO ILLN DRGS Z WVHP. Z HRTM IVZWH":B$="'KIVHHFIV GL
 IVZXGLI.' Z YOFV YFGGLM RH OLXZGVW FMWVI GSV HRTM":E=26:RETURN
320 A$="OZFMXS XLMGILO. GSVIV RH Z WVHP SVIV":S=33:W=37:E=27:RET
URN
330 A$="Z HNZOO XLIIRWLI":N=32:S=34:E=28:RETURN
340 A$="Z HGIZMTV YOFV ILLN. GSVIV RH Z IVW YFGGLM SVIV":N=33:S=
```

```
35:E=29:RETURN
350 A$="Z GRMB HGLIZTV ILLN":N=34:W=40:RETURN
360 A$="Z HNZOO XSZNYVI. GSVIV RH Z DVHG WLLI SVIV":E=30:RETURN
370 A$="Z HNZOO, MZIILI XLIIRWLI":N=42:E=32:RETURN
380 A$="GSV XLNKFGVI XVMGVI. GSVIV RH Z HNZOO HOLG":B$="RM GSV X
LNKFGVI":S=39:RETURN
390 A$="GSV XSVNRXZO OZY":N=38:S=40:RETURN
400 A$="GSV IVZXGLI XLMGILO XVMGVI. GSVIV RH Z YOFV YFGGLM":B$="
ZMW Z IVW LMV. Z HRTM HZBH 'IVZXGLI XLMGILO - IVW=LM, YOFV=LUU":
N=39:E=35:RETURN
410 A$="GSV MFXOVZI IVZXGLI. Z XLNKFGVI IVHGH LM GSV DZOO":E=36:
RETURN
420 A$="GSV DVHG VMW LU Z OLMT XLIIRWLI":S=37:E=43:RETURN
430 A$="GSV VZHG VMW LU Z OLMT XLIIRWLI":W=42:E=44:RETURN
440 A$="Z HVXFIRGB XVMGVI":W=43:E=45:RETURN
450 A$="Z HNZOO OZFMXS TZGV":B$="GSVIV RH Z HNZOO HOLG MVCG GL G
SV OZFMXS TZGV":W=44:RETURN
460 A$="Z HNZOO HKZXVXIZUG. GSVIV RH Z HNZOO HOLG RM GSV NZHGVI"
:B$="OZFMXS XLNKFGVI":W=45
```

Extended descriptions.

```
470 IFA=10AND(D3=1ORD3=2)THENC$="GSV NLMHGVI XZTV RH LKVM"
480 IFA=12ANDD5=0THENC$="GSV XZYRMVG RH OLXPVW"
490 IFA=12ANDD5=1THENC$="GSV XZYRMVG RH LKVM"
500 IFA=20ANDD6=0THENC$="GSV HXIVVM RH YOZMP"
510 IFA=20ANDD6=1THENC$="Z NLERV RH YVRMT KOZBVW LM GSV HXIVVM"
520 IFA=26ANDD9=0THENC$="GSV HZUV RH OLXPVW"
530 IFA=26ANDD9=1THENC$="GSV HZUV RH LKVM"
540 IFA=27ANDE2=0THENC$="GSV XLNKFGVI RH ZXSREV"
550 IFA=27ANDE2=1THENC$="GSV XLNKFGVI RH WVHGILBVW"
560 IFA=36ANDE6=0THENC$="GSV IVZXGLI WLLI RH URINDB OLXPVW"
570 IFA=36ANDE6=1THENC$="GSV IVZXGLI WLLI RH LKVM":W=41
580 IFA=45ANDE9=0THENC$="GSV OZFMXS TZGV RH XOLHVW"
590 IFA=45ANDE9=1THENC$="GSV OZFMXS TZGV RH LKVM":E=46
```

Generate the list of visible items and available exits.

```
600 A$=A$+".":IFLEN(B$)>3THENB$=B$+"."
610 IFLEN(C$)>3THENC$=C$+"."
650 IFN<>0THENE$="MLIGS "
660 IFS<>0THENE$=E$+"HLFGS "
670 IFW<>0THENE$=E$+"DVHG "
680 IFE<>0THENE$=E$+"VZHG "
690 IFE$<>""THENE$=LEFT$(E$,LEN(E$)-1)
```

Describe current location, visible items, and available exits.

```
700 CLS:PRINT"YOU ARE IN: ";P$=A$:GOSUB4:P$=B$:GOSUB4
710 P$=C$:GOSUB4
720 PRINT:PRINT"OBJECTS YOU CAN SEE: ":P$=" ":FORT=1TO16:IFI(T)=
ATHENP$="- "+I$(T):GOSUB4
725 NEXT:IFP$=" "THENP$="-NLGSRMT-":GOSUB4
730 PRINT:PRINT"EXITS: ":P$=E$:GOSUB4
```

Print out additional messages.

```
740 IF(A=40ORA=35ORA=30ORA=31)ANDI(4)=0ANDF3=0THENP$="GSV HNZOO
YOZXP WVERXV RH YORMPRMT":GOSUB4
750 IFA=36ANDI(4)=0ANDF3=0THENP$="GSV HNZOO YOZXP WVERXV RH UOZH
SRMT YIRTSGOB":GOSUB4
760 IFF4<>0THENP$="GSV XLNKFGVI HZBH: "+STR$(F4)+" MINUTES UNTI
L DESTRUCTION":GOSUB4
770 IFD3=1THENP$="* * * ZORVM NLMHGVI ZGGZXPRMT * * *":GOSUB4
780 IFD7=1ORE0=1ORE3=1ORE7=1THENP$="* * * HVXFIRGB KZGILO ZGGZXP
RMT * * *":GOSUB4
```

Get and interpret command.

```
790 PRINT:INPUT"COMMAND";V0$:GOSUB6
800 FORT=1TO4:IFV$=LEFT$(V$(T),1)THENV$=V$(T)
810 NEXTT
820 IFLEN(V$)<3THEN700
830 V1$=LEFT$(V$,3):V2$=RIGHT$(V$,3)
840 FORT=1TO17:IFV1$=LEFT$(V$(T),3)THENV1=T
850 NEXTT:IFV1=0THENP$="R WLM'G FMWVIHGZMW DSZG BLF DZMG NV GL W
L":GOTO8
860 FORT=1TO16:IFV2$=RIGHT$(I$(T),3)THENV2=T
870 NEXTT
880 ONV1GOTO900,940,980,1050,1130,1310,1370,1390,1560,1620,1660,
1790,1860,1910,2010,2170,2240
890 GOTO2250
```

Handle commands.

```
900 IFN=0THEN1110
910 IFD3=1THENP$="GSV NLMHGVI YOLXPH GSV VCRG":GOTO8
920 IFD7=1ORE3=1THENGOTO1100
930 A=N:GOTO2250
940 IFS=0THEN1110
950 IFS=24ANDD8<>0ANDE2<>1THEND7=1:GOSUB1120
960 IFS=30ANDE4<>0ANDE2<>1THENE3=1:GOSUB1120
```

```
970 A=S:GOTO2250
980 IFW=0THEN1110
990 IFD3=1THENP$="GSV NLMHGVI YOLXPH GSV VCRG":GOTO8
1000 IFE0=1ORE3=1ORE7=1THEN1100
1010 IFW=41ANDF3=0THENP$="IZWRZGRLM UILN GSV IVZXGLI SRGH BLF":G
OSUB4:GOTO2510
1020 IFW=30ANDE4<>0ANDE2<>1THENE3=1:GOSUB1120
1030 IFW=27ANDE1<>0ANDE2<>1THENE0=1:GOSUB1120
1040 A=W:GOTO2250
1050 IFE=0THEN1110
1060 IFE0=1ORE3=1ORE7=1THEN1100THEN1100
1070 IFE=27ANDE1<>0ANDE2<>1THENE0=1:GOSUB1120
1080 IFE=44ANDE8<>0ANDE2<>1THENE7=1:GOSUB1120
1090 A=E:GOTO2250
1100 P$="GSV HVXFIRGB ZMWILRW YOLXPH GSV VCRG":GOTO8
1110 P$="GSVIV RH ML DZB GL TL GSZG WRIVXGRLM":GOSUB4:FORJ=1TO10
00:NEXT:GOTO2250
1120 P$="Z HVXFIRGB ZMWILRW ZDZRGH BLF":GOSUB4:RETURN
1130 IFA=1ANDV2$="LXP"THENP$="GSV ZRIOLXP LKVMH ZMW BLF ZIV YOLD
M LFG RMGL GSV EZXFFN LU HKZXV":GOSUB4:GOTO2510
1140 IFA=12ANDV2$="MVG"ANDD5=0ANDI(2)<>0THENP$="BLF ZIVM'G HGILM
T VMLFTS GL ULIXV RG LKVM":GOTO8
1150 IFA=12ANDV2$="MVG"ANDD5=0ANDI(2)=0THENP$="GSV XILDYZI SVOKV
W. GSV XZYRMVG RH MLD LKVM":D5=1:I(5)=ABS(I(5)):GOTO8
1160 IFA=12ANDV2$="MVG"ANDD5=1THENP$="GSV XZYRMVG RH ZOIVZWB LKV
M":GOTO8
1170 IFA=16ANDV2$="VHP"THENP$="LP":I(6)=ABS(I(6)):GOTO8
1180 IFA=17ANDV2$="VHP"THENP$="LP":I(7)=ABS(I(7)):GOTO8
1190 IFA=26ANDV2$="ZUV"ANDD9=1THENP$="GSV HZUV RH ZOIVZWB LKVM":
GOTO8
1200 IFA=26ANDV2$="ZUV"ANDD9=0THENP$="R WLM'G SZEV GSV PVB GL LK
VM GSV HZUV":GOTO8
1210 IFA=31ANDV2$="VHP"THENP$="LP. BLF URMW MLGSRMT RMHRWV":GOTO
9
1220 IFA=32ANDV2$="VHP"THENP$="LP":I(14)=ABS(I(14)):GOTO8
1230 IFA=36ANDV2$="LLI"ANDE6=1THENP$="GSV WLLI RH ZOIVZWB LKVM":
GOTO8
1240 IFA=36ANDV2$="LLI"ANDE6=0ANDI(6)<>0THENP$="BLF WLM'G SZEV G
SV PVB GL GSV WLLI":GOTO8
1250 IFA=36ANDV2$="LLI"ANDE6=0ANDI(6)=0ANDE5=0THENP$="BLF ZIV HF
XPVW RMGL GSV FMKIVHHFIRAVW IVZXGLI YFROWRMT":GOSUB4:GOTO2510
1260 IFA=36ANDV2$="LLI"ANDI(6)=0THENP$="GSV WLLI RH MLD LKVM":E6
=1:GOTO8
1270 IFA=41ANDV2$="MVO"THENP$="GSV XZMVO RH URINOB DLXPVW":GOTO8
1280 IFA=45ANDV2$="LXP"ANDE9=1THENP$="GSV ZRIOLXP RH ZOIVZWB LKV
```

```
M":GOTO8
1290 IFA=45ANDV2$="LXP"ANDE9=0THENP$="GSVIV ZIVM'G ZMB ERHRYOV X
LMGILOH":GOTO8
1300 P$="R XZM'G WL GSZG":GOTO8
1310 IFV2$="GVI"ORV2$="LRW"THENP$="WLM'G YV IRWRXFOLFH":GOTO8
1320 IFV2=0THENP$="R XZM'G WL GSZG":GOTO8
1330 IFI(V2)=0THENP$="BLF ZOIVZWB SZEV GSZG":GOTO8
1340 IFI(V2)<>ATHENP$="R WLM'G HVV RG SVIV":GOTO8
1350 IFP4>=8THENP$="HLIIB, BLF XZM'G XZIIB ZMBGSRMT NLIV":GOTO8
1360 P4=P4+1:I(V2)=0:P$="LP":GOTO8
1370 IFV2=0ORI(V2)<>0THENP$='BLF WLM'G SZEV GSZG":GOTO8
1380 P4=P4-1:I(V2)=A:P$="LP":GOTO8
1390 IF1(5)<>0THENP$="BLF WLM'G SZEV Z DVZKLM":GOTO8
1400 IFA=1ANDV2$="LXP"THENP$="BLF ZIV YOLDM LFG LU GSV ZRIOLXP R
MGL GSV EZXFFN LU HKZXV":GOSUB4:GOTO2510
1410 IFA=27ANDV2$="GVI"THENP$="GSV XLNKFGVI RH WVHGILBVW":E2=1:E
0=0:GOTO8
1420 IFA=38ANDV2$="GVI"THENP$="GSV HSLG IVUOVXGH LUU LU GSV XLNK
FGVI":GOSUB4:GOTO2510
1430 IFA=41ANDV2$="GVI"THENP$="GSV DSLOV MFXOVZI IVZXGLI RH VCKO
LWRMT":GOSUB4:GOTO2510
1440 IFV2$="RWH"ORV2$="YLG"ORV2$="ILO"ORV2$="IWH"ORV2$="ZIW"THEN
V2$="LRW"
1450 IFV2$<>"GVI"ANDV2$<>"LRW"THENP$="GSV OZHVI HSLG SZH ML VUUV
XG":GOTO8
1460 IFV2$="GVI"ANDD3=0THENP$="R WLM'G HVV ZMB NLMHGVI SVIV":GOT
O8
1470 IFV2$="LRW"ANDD7=0ANDE0=0ANDE3=0ANDE7=0THENP$="R WLM'G HVV
ZMB ZMWILRWH SVIV":GOTO8
1480 T=RND(100):IFT>P2+P3+50THENP$="BLF URIV ZMW NRHH":GOTO8
1490 IFD3=1THENP$="BLF SRG GSV NLMHGVI":GOSUB4:D4=D4-((10+P2+P3)
/2):IFD4<=0THEND3=0:D4=0:P$="BLF SZEV PROOVW RG":GOTO8
1500 IFD7=1THENP$="BLF SRG GSV ZMWILRW":GOSUB4:D8=D8-((5+P2+P3)/
2):IFD8<=0THEND7=0:D8=0:P$="RG RH WVHGILBVW":GOTO8
1510 IFE0=1THENP$="BLF SRG GSV ZMWILRW":GOSUB4:E1=E1-((5+P2+P3)/
2):IFE1<=0THENE0=0:E1=0:P$="RG RH WVHGILBVW":GOTO8
1520 IFE3=1THENP$="BLF SRG GSV ZMWILRW":GOSUB4:E4=E4-((5+P2+P3)/
2):IFE4<=0THENE3=0:E4=0:P$="RG RH WVHGILBVW":GOTO8
1530 IFE7=1THENP$="BLF SRG GSV ZMWILRW":GOSUB4:E8=E8-((5+P2+P3)/
2):IFE8<=0THENE7=0:E8=0:P$="RG RH WVHGILBVW":GOTO8
1540 IFD3=1THENP$="RG RH HGROO ZOREV":GOTO8
1550 P$="GSV ZMWILRW RH HGROO UFMXGRLMRMT":GOTO8
1560 IFV2=0THENP$="R XZM'G WL GSZG":GOTO8
1570 IFI(V2)<>0THENP$="R WLM'G SZEV GSZG":GOTO9
1580 IFV2<>9ANDV2<>14THENP$="R XZM'G WL GSZG":GOTO8
```

```
1590 IF(V2=9ANDA=44)OR(V2=14ANDA=38)THENP$="MLGSRMT SZKKVMH":GOT
08
1600 IFV2=9ANDA=38THENF4=35:P$="GSV XLNKFGVI IVKORVH: 'YZHV WVHG
IFXG HVJFVMXV HGZIGVW":GOSUB4:P$="WVHGIFXGRLM RM"+STR$(F4)+" MIN
UTES.'":P4=P4-1:I(9)=100:GOTO8
1610 IFV2=14ANDA=45THENP$="GSV TZGV LKVMH":E9=1:GOTO8
1620 IFV2<>10THENP$="WLM'G YV IRWRXFOLFH":GOTO8
1630 IFI(10)<>0THENP$="BLF WLM'G SZEV GSZG":GOTO8
1640 P$="LP":I(10)=50:P4=P4-1:P1=P1+5+P3:IFP0<P1THENP0=P1
1650 GOTO8
1660 IFA=1ANDV2$="OFV"THENP$="GSV ZRIOLXP LKVMH...":P$="BLF ZIV
YOLDM LFG RMGL GSV EZXFFN LU HKZXV":GOSUB4:GOTO2510
1670 IFA=6ANDV2$="OFV"THENP$="I HGIZMTV, LIZMTV TOLD XLEVIH BLF
ZMW GSVM UZWVH ZDZB":GOTO8
1680 IFA=10ANDV2$="IVW"ANDD3=1THENP$="MLGSRMT SZKKVMH":GOTO8
1690 IFA=10ANDV2$="IVW"THEND3=1:P$="ZM ZORVM NLMHGVI RH IVOVZHVW
. RG RH ZGGIXPRMT BLF!":GOTO8
1700 IFA=13ANDV2$="OFV"THENA=34:P$="Z UOZHS LU ORTSG GVNKLIZIROB
 YORMWH BLF":GOTO8
1710 IFA=20ANDV2$="IVW"ANDD6=0THENP$="MLGSRMT SZKKVMH":GOTO8
1720 IFA=20ANDV2$="IVW"THEND6=0:P$="GSV HXIVVM TLVH YOZMP":GOTO8
1730 IFA=20ANDV2$="OFV"THEND6=1:P$="GSV HXIVVM ORTSGH FK":GOTO8
1740 IFA=31ANDV2$="OFV"THENE5=1:P$="LP":GOTO8
1750 IFA=34ANDV2$="IVW"THENA=13:P$="Z UOZHS LU PRTSG GVNKLIZIROB
 YORMWH BLF":GOTO8
1760 IFA=40ANDV2$="IVW"THENF3=0:P$="LP":GOTO8
1770 IFA=40ANDV2$="OFV"THENF3=1:P$="LP":GOTO8
1780 P$="MLGSRMT SZKKVMH":GOTO8
1790 IFA=22ANDV2$="VVM"THENP$="BLF XZM HVV MLGSRMT LU RMGVIVHG L
M GSV IZWZI":GOTO8
1800 IFV2=0THENP$="R WLM'G SZEV GSZG":GOTO8
1810 IFI(V2)<>0ANDI(V2)<>ATHENP$="R WLM'G SZEV GSZG":GOTO8
1820 IFV2=30RV2=13THENP$="R HVV MLGSRMT HKVXRZO":GOTO8
1830 IFV2=9THENP$="HLIIB, LMOB Z XLNKFGVI XZM IVZW Z KILTIZN":GO
TO8
1840 IFV2=16THENP$="GSV KOZMH ZIV HVZOVW...LMOB XLNNZMW XZM LKVM
 GSVN":GOTO8
1850 P$="R XZM'G IVZW GSZG":GOTO8
1860 CLS:P$="* * * KOZBVI'H RMEVMGLIB * * *":GOSUB4:PRINT
1870 FORI=1TO16:IFI(I)=0THENP$="- "+I$(I):GOSUB4
1880 NEXTI
1890 PRINT:PRINT"HIT ANY KEY TO CONTINUE"
1895 X$=INKEY$:IFX$=""THEN1895
1900 GOTO2400
1910 IFV2=0THENP$="R XZM'G WL GSZG":GOTO8
```

```
1920 IFI(V2)<>0THENP$="R WLM'G SZEV GSZG":GOTO8
1930 IFV2=1ANDA=12AND05=0THENP$="GSV XZYRMVG OLXP RH WVHGILBVW":
D5=1:I(1)=100:I(5)=ABS(I(5)):P4=P4-1:GOTO8
1940 IF(V2=1ORV2=15)AND(D3=1ORD7=1ORE0=1ORE3=1ORE7=1)THENI(V2)=1
00:P4=P4-1:GOTO1490
1950 IF(V2=1ORV2=15)ANDA=1THENP$="GSV ZRIOLXP RH WVHGILBVW...BLF
 ZIV YOLDM LFG RMGL GSV EZXFFN LU HKZXV!":GOSUB4:GOTO2510
1960 IF(V2=1ORV2=15)ANDA=36ANDE6=0ANDE5=0THENP$="GSV WLLI RH WVH
GILBVW...BLF ZIV HFXPVW RMGL GSV MLM-":GOSUB4:P$="KIVHHFIRAVW IV
ZXGLI YFROWRMT":GOSUB4:GOTO2510
1970 IF(V2=1ORV2=15)ANDA=36ANDE6=0ANDF3=0THENP$="GSV WLLI RH WVH
GILBVW. BLF ZIV YLNYZIWVW DRGS IZWRIGRLM":GOSUB4:GOTO2510
1980 IF(V2=1ORV2=15)ANDA=36ANDE6=0THENP$="GSV WLLI RH WVHGILBVW"
:E6=1:I(V2)=100:P4=P4-1:GOTO8
1990 IFV2=1ORV2=15THENP$="GSV "+I$(V2)+" HAS NO EFFECT":I(V2)=10
0:P4=P4-1:GOTO8
2000 GOTO1370
2010 IFV2=0THENP$="R XZM'G WL GSZG":GOTO8
2020 IFI(V2)<>0THENP$="R WLM'G SZEV GSZG":GOTO8
2030 IFV2=5ANDD3=1THENV2$="GVI"
2040 IFV2=5AND(D7=1ORE0=1ORE3=1ORE7=1)THENV2$="LRW"
2050 IFV2=5THEN1390
2060 IFV2=4ANDF3=0AND(A=40ORA=35ORA=30ORA=31)THENP$="GSV YOZXP W
VERXV RH YORMPRMT":GOTO8
2070 IFV2=4ANDF3=0ANDA=36THENP$="GSV YOZXP WVERXV RH UOZHSRMT Y1
RTSGOB":GOTO8
2080 IFV2=4THENP$="GSVIV ZIVM'G ZMB ERHRYOV XLMGILOH LM GSRH WVE
RXV":GOTO8
2090 IFV2=12THENI(12)=A:P4=F4-1:I$(12)="ARMED PHOTON BOMB":F2=35
:P$="GSV KSLGLM YLNY DROO VCKOLWV RM 35 NRMFGVH":GOTO8
2100 IFV2=2ANDA=12AND05=0THEND5=1:P$="GSV XZYRMVG RH MLO LKVM":I
(5)=ABS(I(5)):GOTO8
2110 IFV2=2ANDA=12AND05=1THENP$="GSV XZYRMVG RH ZGIVZWB LKVM":GO
TO8
2120 IFV2=7ANDA=36AND09=0THEND9=1:I(16)=ABS(I(16)):P$="GSV HZUV
LKVMH":GOTO8
2130 IFV2<>11THENP$="DSZG WL BLF DZMG NV GL WL DRGS GSV "+I$(V2)
+"?":GOTO8
2140 IFI(8)<>0THENP$="GSVIV ZIVM'G ZMB YZGGVIRVH ULI GSV IZWRL":
GOTO8
2150 IFF2<>0THENP$="I ELRXV HZBH 'YLNY HGZGFH:"+STR$(F2)+"MINUTE
S UNTIL DETONATION'":GOTO8
2160 P$="GSV IZWRL RH HROVMG":GOTO8
2170 CLS:P$="* * * KOZBVI'H HGZGFH * * *":GOSUB4:PRINT
2180 P$="XFIIVMG SRG KLRMGH ="+STR$(P1):GOSUB4
```

```
2190 P$="WVCGVIRGB ZGGIRYFGV="+STR$(P2):GOSUB4
2200 P$="OFXP ZGGIRYFGV     ="+STR$(P3):GOSUB4
2210 PRINT:PRINT"HIT ANY KEY TO CONTINUE"
2220 X$=INKEY$:IFX$=""THEN2220
2230 GOTO2400
2240 CLS:PRINT"GAME OVER":GOTO 2520
```

Update player status. Conduct compbat if appropriate.

```
2250 IFF2<>0THENF2=F2-1:IFF2<=0THEN2430
2260 IFF4<>0THENF4=F4-1:IFF4<=0THEN2470
2270 IFP1<P0THENP5=P5+.5:IFP5=1THENP5=0:P1=P1+1
2280 IFD3=0ANDD7=0ANDE0=0ANDE3=0ANDE7=0THEN2390
2290 T=RND(100)
2300 IFD3=1THENP$="GSV NLMHGVI ZGGZXPH ":GOSUB4
2310 IFD3<>1THENP$="GSV HVXFIRGB ZMWILRW HSLLGH... ":GOSUB4
2320 IFT>80-(P2+P3)THENP$="RG NRHHVH":GOSUB4:GOTO2390
2330 P1=P1-(RND(5)+RND(5)+RND(5)+RND(5)+15-P3)
2340 IFD3<>1THENP1=P1+5
2350 IFP1<0THEN2510
2380 P$="BLF ZIV SRG!":GOSUB4
2390 IFV1=0ORV1>40R(D3+D7+E0+E3+E7>0)THENFORJ=1TO1000:NEXT
```

Initialize for new turn. Jump to appropriate room description.

```
2400 V$="":V1$="":V2$="":V1=0:V2=0:A$="":B$="":C$="":D$="":E$=""
:N=0:S=0:W=0:E=0
2410 ONAGOSUB10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,
160,170,180,190,200,210,220,230,240,250,260,270,280,290,300,310,
320,330,340,350,360,370,380,390,400,410,420,430,440,450,2550:GOT
0470
2420 GOTO10
```

Evaluate end-game conditions and display appropriate messages.

```
2430 CLS:IFA=46THENF4=-1:GOTO2550
2440 P$="GSV KSLGLM YLNY VCKOLWVH...GSV VMGRIV XLNKOVC RH WVHGIL
BVW":GOSUB4
2450 P$="BLF SZEV YVVM PROOVW YB GSV ULIXV LU GSV YOZHG":GOSUB4
2460 PRINT:PRINT:GOTO2510
2470 CLS:IFA=46THEN2550
2480 IFA=38THENP$="GSV XLNKFGVI UOZHSVH YIRTSGOB, VNRGGRMT HKZIP
H RM ZOO WRIVXGRLMH":GOSUB4
2490 P$="GSV XLNKOVC HFWWVMOB VCKOLWVH RMGL NROORLMH LU KRVXVH":
GOSUB4
```

```
2500 P$="BLF ZIV PRODVW YB GSV UZODRMT WVYIRH ZILFMW BLF":GOSUB4
:PRINT:PRINT
2510 P$="BLF ZIV WVZW!":GOSUB4
2520 INPUT'DO YOU WANT TO PLAY AGAIN";A$
2530 IFLEFT$(A$,1)="Y"THEN2610
2540 CLS:END
2550 P$="GSV HKZXV HSRK HFWWVMOB ORUGH RMGL LIYRG ZILFMW GSV KOZ
MVG":GOSUB4
2560 IF((F2=0)OR(F2<>0ANDI(12)<>41))ANDF4=0THENP$="BLF WRWM'G WV
HGILB GSV YZHV. NRHHRLM UZROVW":GOSUB4:PRINT:GOTO2520
2570 P$="UILN Z WRHGZMXV, BLF XZM HVV GSV ZORVM YZHV VCKOLWV":GO
SUB4
2580 IFI(16)<>0THENP$="BLF WRWM'G IVXLEVI GSV HVXIVG KOZMH MVVWV
W YB HGZI XLNNZMW":GOSUB4:PRINT:GOTO2520
2590 P$="NRHHRLM RH Z HFXXVHH!":GOSUB4
2600 GOTO2520
```

Initialize workspace. Read in items and verbs.

```
2610 CLEAR300:DIMI$(16),I(16),V$(17):C155=-155:C64=64:N1=1
2620 CLS
2630 PRINTTAB(15)"OPERATION: SABOTAGE BY RAY SATO"
2640 FORT=1TO16:READI$(T),I(T):NEXT
2660 FORT=1TO17:READV$(T):NEXT
2690 FORT=1TO40:P0=P0+RND(2):NEXTT
```

Establish player-attribute points. Jump to first room.

```
2700 P1=P0
2710 FORT=1TO10:P2=P2+RND(2):NEXTT
2720 FORT=1TO10:P3=P3+RND(2):NEXTT
2730 FORT=1TO50:D4=D4+RND(2):D8=D8+RND(2):E1=E1+RND(2):E4=E4+RND
(2):E8=E8+RND(2):NEXTT
2740 A=1:P1=1
2750 GOSUB10:GOTO470
```

Item and verb data.

```
2760 DATAKOZHGRX VCKOLHREV,0,XILDYZI,7,XZOVMWZI,8,HNZOO YOZXP WV
ERXV,9,OZHVI KRHGLO,-12,HVXFIRGB PVB,-16,VOVXGILMRX XLMGILO YZGL
M,-17,YZGGVIRVH,18,XLNKFGVI WVHGIFYG KILTIZN,21,HROEVI KROO,23,K
LIGZYOV IZWRL,25,OZITV KSLGLM YLNY,28
2770 DATATZOZXGRX XSZIG,32,OZFMXS HBHGVN XZHHV3GV,-32,MRGILTOBXV
IRM,39,HVXIVG KOZMH,-26
2780 DATAMLIGS,HLFGS,DVHG,VZHG,LKVM,TVG,WILK,HSLLG,RMHVIG,VZG,KF
HS,IVZW,RMEVMGLIB,GSILD,FHV,HGZGFH,JFRG
```

TRS-80® SWAT TABLE FOR: OPERATION SABOTAGE

(Modified Parameters: NU = 5 B = 200)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
2 - 10	XQ	238	1410 - 1430	XH	209
20 - 60	SU	221	1440 - 1470	QG	259
70 - 100	MX	222	1480 - 1500	QP	253
110 - 140	HV	221	1510 - 1530	NA	297
150 - 180	CR	228	1540 - 1580	QP	197
190 - 210	DI	213	1590 - 1610	RA	240
220 - 250	XZ	207	1620 - 1660	NP	244
260 - 280	LY	225	1670 - 1690	RG	214
290 - 310	SE	207	1700 - 1730	QS	231
320 - 350	RL	221	1740 - 1780	HC	209
360 - 390	NU	223	1790 - 1820	KF	204
400 - 410	LS	206	1830 - 1860	WX	206
420 - 450	XG	232	1870 - 1900	FU	108
460 - 490	BT	225	1910 - 1940	IL	236
500 - 540	JL	217	1950 - 1960	BH	244
550 - 590	HK	245	1970 - 1990	HY	255
600 - 670	RV	127	2000 - 2040	XW	151
680 - 720	SS	177	2050 - 2080	GF	222
725 - 750	GF	214	2090 - 2110	HA	237
760 - 780	JF	201	2120 - 2150	BR	272
790 - 830	NU	111	2160 - 2200	UL	193
840 - 880	DT	226	2210 - 2250	SB	121
890 - 930	IZ	105	2260 - 2300	LD	157
940 - 980	MB	108	2310 - 2350	LT	165
990 - 1030	ZL	202	2380 - 2410	YE	339
1040 - 1080	RC	124	2420 - 2460	IW	176
1090 - 1130	IW	272	2470 - 2500	ZN	230
1140 - 1160	JR	246	2510 - 2550	UY	163
1170 - 1200	ZD	217	2560 - 2580	LY	250
1210 - 1240	IC	236	2590 - 2630	GB	145
1250 - 1270	TU	222	2640 - 2710	QD	109
1280 - 1310	KN	209	2720 - 2760	ZZ	348
1320 - 1360	LF	206	2770 - 2780	AZ	184
1370 - 1400	YV	214			

NOTES:

BROADWAY

by Robert Saturn

***Broadway* is a simulation for a TRS-80 with 16K RAM.**

This simulation begins with a flashing Broadway marquee. The player (henceforth called the producer) is then asked to name the production company. This name will appear on the weekly report and on the closing notice at the end of the simulation. The program will accept any string (no commas, no double quotes) up to 15 characters and spaces.

After a basic introduction and some instructions, the producer attempts to raise $1,000,000 to produce the show. As in the real world, past performance (as shown by a randomly generated "track record") controls the ease with which the money can be raised. The names entered in this section have no bearing on the amount of money raised (that is controlled by a random number in conjunction with the "track record"), but does add to the fun of the simulation when the names of friends and realtives are used. If the money is not raised within eight tries, the simulation ends, and the producer can try again with a new "track record." If the money is raised, it will be more than the requisite $1,000,000. The method used to lose the excess is as true-to-life as any.

The program then tells the producer about some assumed payroll expenses that will be deducted each week, and then lets the producer hire one of three people for each of ten key jobs. As each person is hired, his respective fee is added to a running total that will be deducted before opening night, and his weekly salary is added to a running total that is deducted from the production company's funds each week. Each choice also assigns a certain number of quality points to yet another running total. These point values reflect the relative importance of each job. Choosing a high-priced employee tends to result in higher quality, but too large a payroll can bankrupt the producer. A #1 employee raises the quality point total, a #2 employee leaves it unchanged, and a #3 employee reduces the quality point total.

Next, a theatre must be chosen. The larger the theatre (with a larger potential gross), the greater the expenditures both in rental and the cost of the staff that the theatre employs.

Each of the producer's four designers has submitted three designs in his respective department. Each design differs in quality, and therefore in cost. The one-time fee (for items purchased) and the weekly cost (for items rented) will be added to the applicable totals on opening night. Technical appearance quality points are compiled as each selection is made.

The rehearsal period is five weeks. The weekly payroll total is deducted for five weeks before opening. The one-time fees are deducted after the rehearsal period is over. At this point, the director has the option of having further rehearsals (a random function). Each extra week of rehearsals will cost the producer one week's payroll. Here's where a high payroll and high technical expenses coupled with a few extra weeks of rehearsal can put the producer out of business.

On opening night, the producer will get a report of the total weekly cost figures. These figures, of course, are minimums. Salary changes and advertising will add to the weekly costs.

The reviews are generated as follows: each of the five reviewers has five reviews upon which to draw. They range from great to very poor. (Simon of *New York Magazine* will always give a bad review — a little humor for those familiar with the New York theatre scene.) The quality points earned for personnel are doubled; the quality points earned for technical appearance are added; and the total is multiplied by a random number ranging from 1 to 10 (this simulates the reviewer's mood). This process determines which of the five reviews will be selected, and is repeated for each of the five reviewers. Thus five different people can see the same show, yet write five different review, a frequent occurrence on Broadway. As each review appears, box office points are earned. Better reviews earn more points. More points mean more tickets are sold. Box office points are also affected by advertising (1 point for each $10,000 spent per week), ticket price (high prices keep people away), aging of the show (the longer a show runs, the harder it is to get an audience), and random events that will be explained later.

After seeing the reviews, the producer has the option of closing the show, and returning what remains of the original $1,000,000 to the investors. Also, the producer may exercise this option after getting the report each week.

If the show remains open, the producer must decide how much to spend on advertising each week. Any amount down to zero may be spent, available funds after payroll expenses being the only limitation. The money used for advertising becomes part of the weekly expense only for the week in which it was allocated. Use no dollar signs or commas when you tell the computer how much to spend on advertising.

The current ticket price then appears on the screen, and can be increased or decreased any whole number of dollars.

At this point, a random event may occur (usually every three or four weeks). There are both good events and bad events, and all will affect the box office action in one way or another. The two events dealing with pay raises will increase your weekly payroll. The other events will increase or decrease your current total of box office points.

Then the weekly report will appear. The number of tickets sold for the week is figured, using box office points, the original quality points of the show (before the critics influenced the public), a random number (to simulate the public mood), and the seating capacity of the theatre. The number of tickets sold is then multiplied by the current ticket price, and the gross is displayed. Then the expenses for the week are displayed, and deducted from the gross, yielding the net profit. Any loss is deducted from previous profits. If there are no previous profits, the loss is deducted from what remains of the original $1,000,000. If the front money is exhausted, the producer is out of business. In addition, every 13 weeks, a quarterly payment of 98% of the current profits is distributed to the investors, and will not be available for advertising or covering losses.

The show may be closed after the report for any week is displayed. When the producer closes the show, the closing notice is posted, the salaries for the final two weeks are paid by the salary bonds that were posted before opening, and the final totals are displayed, showing, among other other things, the percentage of return to the investors on their investments.

```
                SS SS SS SS SS SS SS SS SS SS SS
                SS                          SS
                SS       TRS-80 BASIC       SS
                SS        'Broadway'        SS
                SS   Author:  Robert Saturn  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 REM COPYRIGHT (C) 1979 ROBERT SATURN
15 CLEAR 1000
20 GOSUB100
30 GOTO220
100 A$="¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥"+CHR$(13)+"    B R O A D W A Y
     "+CHR$(13)+"¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥"
110 B$=" ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ "+CHR$(13)+"¥   B R O A D W A Y
     ¥"+CHR$(13)+" ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ "
120 CLS:PRINT CHR$(23)
130 FOR ZA=1TO10
140 PRINT@128,A$
150 FORZB=1TO100:NEXT
160 PRINT@128,B$
170 FORZB=1TO100:NEXT
180 NEXTZA
190 RETURN
220 DEFINTA-Z:DEFSNGA,F,I,T,P,S:RANDOM:IT=1:AT=16:IU=1:SP=16:TT=
10000
230 ON ERROR GOTO3190
250 CLS:PRINT@256,"THIS PRODUCTION COMPANY WILL BE KNOWN AS ";ST
RING$(10,95);:INPUT" PRODUCTIONS";PR$
310 GOSUB120:PRINT@448,"BREAK A LEG & BRING IN A HIT"
320 GOSUB130:PRINT@580,"BUT REMEMBER....."
330 GOSUB130:PRINT@704,"THERE'S A BROKEN HEART ";:GOSUB130:PRINT
@770,"(AND BANK ACCOUNT) ";:GOSUB130:PRINT@836,"FOR EVERY LIGHT
ON BROADWAY.":GOSUB130
340 CLS:FM=1000000:X=10000:R=RND(100):TR=R/50:P$="$$#######,#"
360 PRINT"FOR THE PURPOSES OF THIS SIMULATION, YOU WILL GET 8 CH
ANCES":PRINT"TO RAISE A TOTAL OF $ 1,000,000 TO FINANCE YOUR SHO
W."
380 PRINT"YOUR TRACK RECORD ON PREVIOUS SHOWS IS";R;"%. THIS WIL
L":PRINT"DETERMINE HOW EASY IT IS FOR YOU TO RAISE MONEY.":AI=0:
C=1
390 PRINT:PRINT"INVESTOR #";C
400 INPUT"WHO WILL YOU ASK (TYPE THE NAME)";I$:CLS:I=RND(25)*TR
410 IFI<2PRINTI$;" SAID, 'I'M SORRY BUT, NO.'";:PRINT:I=0:GOTO520
```

```
420 IFI<5PRINTI$;" THINKS YOU'RE CRAZY,":PRINT" BUT WILL INVEST"
;:GOSUB690:GOTO520
430 IFI<7PRINTI$;" CAN'T REALLY AFFORD MUCH BUT,":PRINT" WANTS T
O HELP WITH";:GOSUB690:GOTO520
440 IFI<10PRINTI$;" NEEDS A TAX WRITE-OFF AND WILL INVEST";:GOSU
B690:GOTO520
450 IFI<12PRINTI$;" WHO IS STILL WORKING ON THAT FIRST":PRINT" M
ILLION, INVESTS";:GOSUB690:GOTO520
460 IFI<15PRINTI$;" (WHO INVESTS IN ANYTHING) THINKS":PRINT" YOU
 HAVE A WINNER AND INVESTS";:GOSUB690:GOTO520
470 IFI<17PRINTI$;" LOVES TO THROW MONEY AWAY,":PRINT" AND INVES
TS";:GOSUB690:GOTO520
480 IFI<20PRINTI$;" (WHO HAS NO TASTE) HAS FAITH IN YOU":PRINT"
AND YOUR SHOW AND INVESTS";:GOSUB690:GOTO520
490 IFI<22PRINTI$;" WHO HAS TURNED DOWN EVERY OTHER":PRINT" PROD
UCER ON BROADWAY SAYS 'YES' AND INVESTS";:GOSUB690:GOTO520
500 IFI<24PRINTI$;", A NOTED PATRON OF THE ARTS, INVESTS";:GOSUB
690:GOTO520
510 IFI>=24PRINTI$;" WHO HAS MORE MONEY THAN GOD, INVESTS";:GOSU
B690
520 PRINT:PRINT"SO FAR YOU HAVE RAISED ";
530 PRINT USINGP$;AI
540 IF AI<FM THEN C=C+1:GOTO580
550 IFAI>=FMPRINT:PRINT"THAT'S";
560 PRINTUSINGP$;AI-FM;:PRINT" EXTRA !":GOTO610
580 IF C=9THENFORZA=1TO500:NEXT:GOSUB120:PRINT:PRINT"SORRY, YOU
COULDN'T RAISE THE MONEY.":GOSUB130:PRINT@512,"THAT'S SHOW BIZ!!
!!":PRINT:PRINT:INPUT"WANT TO TRY AGAIN";TA$:IFLEFT$(TA$,1)="Y"T
HEN340ELSEEND
590 PRINT"YOU STILL NEED ";
600 PRINT USING P$;FM-AI:GOTO390
610 FORB=1TO3000:NEXT:C=0
620 GOSUB120
630 PRINT@450,"YOU HAVE RAISED THE MONEY"
640 GOSUB130
660 CLS:PRINT"YOUR GENERAL MANAGER WILL STEAL THE EXTRA";
670 PRINT USING P$;(AI-FM)
680 GOTO730
690 I1=I*X
700 PRINT:PRINT USING P$;I1
710 AI=AI+I1:RETURN
730 FORB=1TO5000:NEXT:CLS
740 PRINT"NOW THAT YOU HAVE RAISED THE MONEY, YOU MUST DECIDE HO
W":PRINT"TO SPEND IT. FOR EACH OF THE MAJOR EXPENSES PRESENTED T
O YOU,"
```

```
745 PRINT"DECIDE ON HOW MUCH YOU WILL SPEND INITIALLY AND WEEKLY
. DURING":PRINT"THE FIRST FIVE WEEKS (WHILE YOU ARE REHEARSING)
THERE WILL"
750 PRINT"BE NO INCOME, SO MAKE SURE THAT YOUR EXPENSES DO NOT":
PRINT"EXCEED $1,000,000 OR YOU'LL HAVE NOTHING LEFT FOR DELAYED"
:PRINT"OPENINGS, ADVERTISING, ABSORBING A LOSS FOR A WEEK OR TWO
, AND"
755 PRINT"THE FACT THAT A FULL HOUSE EVERY NIGHT WILL ONLY GROSS
 BETWEEN"
760 PRINT"$150,000 AND $225,000. YOUR OBJECT OF COURSE, IS TO SH
OW":PRINT"A PROFIT EACH WEEK. REMEMBER, OUT OF YOUR 'FRONT MONEY
' YOU":PRINT"MUST PAY ALL OF YOUR PRE-OPENING EXPENSES (E.G. SET
S, COSTUMES,":PRINT"ETC.)"
770 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE";E:CLS:PRINT"FIRST YOU
 MUST HIRE A COMPANY.":PRINT"CERTAIN WEEKLY PAYROLL EXPENSES ARE
 ASSUMED:":PRINT"$ 5,000 FOR CONTRACT STAGEHANDS":PRINT"$ 3,000
FOR WARDROBE DEPT."
780 PRINT"$ 3,000 FOR CONTRACT MUSICIANS":PRINT"$   750 FOR PRES
S REPS.":PRINT"$ 6,000 FOR MANAGEMENT STAFF":PRINT"$ 2,500 FOR S
TAGE MANAGEMENT":PRINT"$17,500 FOR NON-STARING ACTORS":PRINT"---
-----":PRINT"$37,950":P=37950
810 PRINT:PRINT"WE WILL NOW DEAL WITH THE SALARIES THAT CAN BE N
EGOTIATED.":INPUT"PRESS 'ENTER' TO CONTINUE";E
820 CLS:PRINT"FOR EACH JOB YOU WILL BE GIVEN 3 ALTERNATIVES.":PR
INT"IN GENERAL, THE MORE YOU SPEND, THE HIGHER THE CALIBER OF"
825 PRINT"PERSON YOU WILL HIRE AND THE BETTER THE RESULTS (MAYBE
 ).":PRINT"OF COURSE, THE MORE YOU SPEND, THE HIGHER YOUR WEEKLY
PAYROLL":PRINT"WILL BE. SPEND";
830 PRINT" MONEY WHERE YOU THINK IT'S IMPORTANT.":PRINT:INPUT"PR
ESS 'ENTER' TO CONTINUE";E
850 FORC=1TO12:CLS:READJ$,F1,W1,F2,W2,F3,W3,I(1),I(2),I(3):PRINT
"JOB - ";J$:PRINT:PRINT:PRINTTAB(8)"INITIAL FEE";TAB(30)"WEEKLY
ROYALTY (OR SALARY)"
860 PRINT:PRINTTAB(0)"1";TAB(12)F1;TAB(35)W1
870 PRINTTAB(0)"2";TAB(12)F2;TAB(35)W2
880 PRINTTAB(0)"3";TAB(12)F3;TAB(35)W3
890 PRINT:PRINT"WHICH ";J$;" WILL YOU HIRE (1, 2, OR 3)":INPUTH
900 IFH=1THENPX=PX+F1:P=P+W1:GOTO940
910 IFH=2THENPX=PX+F2:P=P+W2:GOTO940
920 IFH=3THENPX=PX+F3:P=P+W2:GOTO940ELSE890
940 IT=IT*I(H)
950 NEXT
970 IT=IT/1000
990 IFIT>500THENIT=2:GOTO1060
1000 IFIT>200THENIT=1.5:GOTO1060
```

```
1010 IFIT>70THENIT=1.1:GOTO1060
1020 IFIT<.1THENIT=.5:GOTO1060
1030 IFIT<1THENIT=.75:GOTO1060
1040 IT=1
1060 CLS:PRINT"YOU NOW HAVE A COMPANY TO WORK WITH.":PRINT"NOW D
ECIDE ON YOUR OTHER EXPENSES.":PRINT:PRINT"THE MOST IMPORTANT IS
 THE THEATRE. AGAIN WE HAVE A CHOICE":PRINT"OF THREE. THEY DIFFE
R IN CAPACITY AND THEREFORE IN COST."
1070 PRINT"ASSUME A TICKET PRICE OF $16 AT THIS TIME.":PRINT:PRI
NTTAB(0)"THEATRE";TAB(9)"CAPACITY";TAB(22)"POSSIBLE WEEKLY GROSS
";TAB(47)"COST (% OF GROSS"
1080 PRINTTAB(49)"+HOUSE PAYROLL)"
1090 PRINTTAB(0)"1";TAB(10)"1400";TAB(23)"179,200";TAB(48)"22%+$
19,000":PRINTTAB(0)"2";TAB(10)"1500";TAB(23)"192,000";TAB(48)"25
%+$20,000":PRINTTAB(0)"3";TAB(10)"1600";TAB(23)"204,800";TAB(48)
"28%+$21,000"
1100 PRINT:INPUT"WHICH THEATRE (1, 2, OR 3)";H
1110 IFH=1THENCP=1400:TX=.22:PR=19000:GOTO1150
1120 IFH=2THENCP=1500:TX=.25:PR=20000:GOTO1150
1130 IFH=3THENCP=1600:TX=.28:PR=21000:GOTO1150ELSE1100
1140 'CHOOSE DESIGN VERSION FOR EACH DEPARTMENT - J$=DEPARTMENT,
 F=ONE-TIME FEE, W=WEEKLY FEE, I=TECHNICAL QUALITY POINTS
1150 CLS:PRINT"NOW DETERMINE YOUR OTHER PRE-OPENING EXPENSES."
1160 FORC=1TO5:READJ$,F1,W1,F2,W2,F3,W3,I(1),I(2),I(3)
1170 PRINT:PRINT"EXPENSE - ";J$
1180 PRINT:PRINTTAB(15)"INITIAL";TAB(30)"WEEKLY"
1190 PRINTTAB(0)"1";TAB(16)F1;TAB(31)W1
1200 PRINTTAB(0)"2";TAB(16)F2;TAB(31)W2
1210 PRINTTAB(0)"3";TAB(16)F3;TAB(31)W3
1220 PRINT:INPUT"WHICH DESIGN VERSION WILL YOU USE";H
1230 IFH=1THENPX=PX+F1:PY=PY+W1:GOTO1270
1240 IFH=2THENPX=PX+F2:PY=PY+W2:GOTO1270
1250 IFH=3THENPX=PX+F3:PY=PY+W3:GOTO1270ELSE1220
1270 IU=IU*I(H)
1280 CLS:NEXT
1300 IFIU>7THENIU=2:GOTO1330
1310 IFIU>1THENIU=1.5:GOTO1330
1320 IFIU<1THENIU=.8
1330 CLS:PRINT"THE REST OF THE PRE-OPENING EXPENSES ARE AS FOLLO
WS:":PRINT:PRINT"TAKE-IN AND SET-UP OF SHOW  $ 33,000":PRINT"PRE
-OPENING ADVERTISING";CHR$(199);"30,000";PRINT"OPENING NIGHT PAR
TY";CHR$(203);" 5,000":PRINT"TRAVEL EXPENSES";CHR$(207);" 3,000
"
1340 PRINT"TICKET PRINTING";CHR$(207);" 1,500":PRINT"REHEARSAL S
PACE";CHR$(207);" 3,000":PRINT"SALARY BONDS"CHR$(210);"10,000":P
```

```
RINT"";TAB(28)"----------":PRINT"";TAB(28)"$ 85,500":PRINT:INPUT
"PRESS 'ENTER' TO CONTINUE";E:PX=PX+85500
1360 CLS:PRINT"YOU ARE NOW READY TO BEGIN THE FIVE WEEK REHEARSA
L PERIOD. YOUR":PRINT"PAYROLL EXPENSES WILL BE DEDUCTED AFTER EA
CH WEEK OF"
1365 PRINT"REHEARSALS. AFTER THE LAST WEEK, THE OTHER PRE-OPENIN
G EXPENSES":PRINT"WILL BE DEDUCTED.":PRINT:INPUT"PRESS 'ENTER' T
O CONTINUE";E
1370 FORW=1TO5:GOSUB120:PRINT:PRINT"END OF WEEK";W;"OF REHERSALS
"
1380 PRINT:PRINT"EXPENSES THIS WEEK -";:PRINTUSINGP$;P
1390 FM=FM-P:PRINT:IFFM<0GOTO2780
1400 PRINT"LEFT FROM FRONT MONEY";:PRINTUSINGP$;FM:GOSUB130:NEXT
1420 FM=FM-PX:IFFM<0GOTO2780
1430 CLS:PRINT"NOW THAT REHERSALS ARE OVER AND YOUR PRE-OPENING"
:PRINT"EXPENSES ARE PAID, YOU HAVE";
1440 PRINTUSINGP$;FM;:PRINT" LEFT IN AN":PRINT"EMERGENCY ACCOUNT
.":PRINT:INPUT"PRESS 'ENTER' TO CONTINUE";E
1460 CLS:R=RND(10):IFR<6PRINT"THE DIRECTOR WANTS ANOTHER WEEK OF
 REHERSALS BEFORE OPENING"ELSEGOTO1540
1470 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE";E
1490 FM=FM-P:IFFM<0GOTO2780
1500 PRINT:PRINT"YOU NOW HAVE";
1510 PRINTUSINGP$;FM;:PRINT" LEFT IN EMERGENCY RESERVE."
1520 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE";E:GOTO1460
1540 FORJ=1TO4:CLS:PRINTCHR$(23):PRINT@268,"I T ' S":PRINT@402,"
O P E N I N G":PRINT@540,"N I G H T":FORB=1TO200:NEXT:CLS:FORB=1
TO100:NEXT:NEXT:P=P+PY
1550 PRINT"WELL, HERE WE GO. LET'S HAVE A DRINK BEFORE THE REVIE
WS COME IN."
1560 PRINT:PRINT"OH BY THE WAY, YOUR REGULAR WEEKLY EXPENSES (IN
CLUDING"
1570 PRINT" THE WEEKLY TECHNICAL EXPENSES THAT BEGIN TONIGHT) AN
D"
1580 PRINT"THE THEATRE RENTAL ARE :"
1600 P=P+PR
1610 PRINTUSINGP$;P;:PRINT" PLUS";TX*100;"% OF THE GROSS."
1620 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE";EN
1640 CLS:PRINT"HERE COMES THE PRESS AGENT WITH THE REVIEWS"
1660 IV=IT*IT+IU
1670 IFIV>5IX=2:GOTO1720
1680 IFIV>3THENIX=1.5:GOTO1720
1690 IFIV>2THENIX=1.25:GOTO1720
1700 IFIV>1THENIX=1.1:GOTO1720
1710 IFIV<=1THENIX=1
```

```
1720 GOSUB2830:PRINT:PRINT"KERR - THE TIMES- ";:ONR1GOTO1740,175
0,1760,1770,1780,1780
1740 PRINT"THE WORST THING I EVER SAW'":P9=P9+1:GOTO1790
1750 PRINT"'WEAK SHOW: POOR ACTING, POOR DIRECTING POOR,";CHR$(2
13);"POOR INVESTORS'":P9=P9+2:GOTO1790
1760 PRINT"'I LIKED IT - WITH RESERVATIONS'":P9=P9+3:GOTO1790
1770 PRINT"'A THROUGHLLY ENJOYABLE EVENING'":P9=P9+4:GOTO1790
1780 PRINT"'A FINE PIECE OF THEATRE'":P9=P9+5
1790 GOSUB2830:PRINT"BARNES - THE POST -";:ONR1GOTO1800,1810,182
0,1830,1840,1840
1800 PRINT"'A TERRIBLE SHOW'":P9=P9+1:GOTO1850
1810 PRINT"'I WAS BORED DURING TH 1ST ACT AND GONE";CHR$(217);"D
URING THE 2ND'":P9=P9+2:GOTO1850
1820 PRINT"'A PLEASENT EVENING'":P9=P9+3:GOTO1850
1830 PRINT"'MUCH FUN - GO SEE IT.'":P9=P9+4:GOTO1850
1840 PRINT"'A MUST SEE! RUN, DO NOT WALK TO THIS.'":P9=P9+5:GOTO
1850
1850 GOSUB2830:PRINT"WATT - THE NEWS -";:ONR1GOTO1860,1870,1880,
1890,1900
1860 PRINT"'THE WORST PLAY IN MY 80 YEARS OF REVIEWING'":P9=P9+1
:GOTO1910
1870 PRINT"'THE LAST PLAY I SAW THAT WAS THIS BAD WAS IN";CHR$(2
12);"1904":P9=P9+2:GOTO1910
1880 PRINT"'NICE. BROUGHT BACK MEMORIES OF";CHR$(226);"-THE BLAC
K CROOK-'":P9=P9+3:GOTO1910
1890 PRINT"'FUN FOR ALL AGES. BRING YOUR GRANDPARENTS'":P9=P9+4:
GOTO1910
1900 PRINT"'THE BEST THING I'VE SEEN THIS CENTURY'":P9=P9+5
1910 GOSUB2830:PRINT"SHALIT - NBC NEWS -";:ONR1GOTO1920,1930,194
0,1950,1960,1960
1920 PRINT"'SHOULD NOT BE ALLOWED ANY CLOSER TO NEW YORK";CHR$(2
12);"THAN BOISE, IDAHO.'":P9=P9+1:GOTO1970
1930 PRINT"'I'VE SEEN WORSE, BUT NOT MANY'":P9=P9+2:GOTO1970
1940 PRINT"'IT WON'T RUN LONGER THAN -DOLLY- BUT MAYBE";CHR$(216
);"LONGER THAN -KELLY-'":P9=P9+3:GOTO6025
1950 PRINT"'A HOOT'":P9=P9+4:GOTO1970
1960 PRINT"'IF YOU HURRY YOU CAN GET TICKETS FOR LATE IN 1982.":
P9=P9+5
1970 GOSUB2830:PRINT"SIMON - NEW YORK MAG -";:ONR1GOTO1980,1990,
2000,2010,2020,2020
1980 PRINT"'I HATED IT'":P9=P9+1:GOTO2040
1990 PRINT"'A TERRIBLE SHOW'":P9=P9+2:GOTO2040
2000 PRINT"'IT'S THE PITS'":P9=P9+3:GOTO2040
2010 PRINT"'THE WORST THIS YEAR'":P9=P9+4:GOTO2040
2020 PRINT"'I COULDN'T STAND IT'":P9=P9+5
```

```
2030 'KEEP THE SHOW OPEN AFTER THE REVIEWS?
2040 GOSUB2850
2050 CLS:PRINT"ALL RIGHT, THE SHOW IS OPENED.":PRINT"AFTER EACH
WEEK YOU WILL GET A REPORT AS TO HOW"
2055 PRINT"MANY TICKETS ARE SOLD AND HOW MUCH MONEY WAS TAKEN IN
. YOU":PRINT"WILL ALSO BE ASKED HOW MUCH YOU WANT TO SPEND ON AD
VERTISING":PRINT"AND IF YOU WANT TO CHANGE THE TICKET PRICE."
2060 PRINT"REMEMBER, CHEAPER PRICES = HIGHER VOLUME.":PRINT:INPU
T"PRESS 'ENTER TO CONTINUE";E:WK=0
2070 PRINT:PRINT"YOU HAVE";USINGP$;FM;:PRINT" LEFT IN FRONT MONE
Y ACCOUNT"
2090 GOSUB2880
2110 GOSUB2900
2120 WK=WK+1:GOSUB120
2140 IFWK/3=INT(WK/3)THENP9=P9-2
2150 IFWK/10=INT(WK/10)THENP9=P9-2
2160 IFWK/18=INT(WK/18)THENP9=P9-2
2180 IFWK/13=INT(WK/13)GOSUB2980
2200 P9=P9+AD/TT
2220 R=RND(10):IFR>6THENR=RND(12):PRINT:ONRGOTO2240,2250,2260,22
70,2280,2290,2300,2310,2320,2330,2340,2350
2230 GOTO2380
2240 PRINT"STAR GETS SICK - MUST BE REPLACED":P9=P9-2:GOTO2360
2250 PRINT"UNIONS GET PAY HIKE":P=P+(P*.03):P=INT(P):GOTO2360
2260 PRINT"STAR'S CONTRACT ENDS-WANTS MORE $$$":P=P+(P*.03):P=IN
T(P):GOTO2360
2270 PRINT"SHOW WINS 3 TONYS":P9=P9+2:GOTO2360
2280 PRINT"FREE PUBLICITY DUE TO GOOD PRESS AGENT":P9=P9+1:GOTO2
360
2290 PRINT"SHOW WINS CRITIC CIRCLE AWARD":P9=P9+1:GOTO2360
2300 PRINT"HEAVY SNOW FOR 5 WEEKS":P9=P9-1:GOTO2360
2310 PRINT"NEWSPAPER STRIKE IN IT'S 2ND MONTH":P9=P9-1:GOTO2360
2320 PRINT"T.V. REPORTS ON HIGH CRIME IN THE BROADWAY AREA":P9=P
9-1:GOTO2360
2330 PRINT"CITY REPORTS CONVENTION BUSINESS UP":P9=P9+1:GOTO2360
2340 PRINT"GOOD WORD OF MOUTH ON THE SHOW":P9=P9+1:GOTO2360
2350 PRINT"BAD WORD OF MOUTH ON THE SHOW":P9=P9-1:GOTO2360
2360 FORB=1TO2000:NEXT
2380 CLS:PRINTPR$;" PRODUCTIONS";CHR$(201);"WEEKLY REPORT":PRINT
"WEEK";WK;CHR$(210);WK*8;"PERFORMANCES":PRINT:PRINT"THEATRE CAPA
CITY (PER PERF)";CP
2390 PRINT"TICKET PRICE";:PRINTUSINGP$;SP
2400 PRINT"MAXIMUM GROSS (PER WEEK)";:PRINTUSINGP$;SP*CP*8:IFP9<
0THENP9=1
2410 PRINT:PRINT"TICKETS SOLD THIS WEEK";:TS=0:TS=((IX*P9)/60)*C
```

```
P#8:IFTS>CP#8THENTS=CP#8
2420 PRINTINT(TS)
2430 PRINT"GROSS THIS WEEK";:TG=TS#SP:PRINTUSINGP$;TG
2440 PRINT"EXPENSES THIS WEEK";:P5=P+(TG#TX)+AD:PRINTUSINGP$;P5
2450 PRINT:ST=TG-P5:PRINT"NET ";:IFST<0PRINT"LOSS ";
2460 IFST>0PRINT"PROFIT ";
2470 PRINT"THIS WEEK ";USINGP$;ST
2480 IFST<0THENSD=SD-ABS(ST)
2490 IFSD<0THENFM=FM-ABS(SD):SD=0
2500 IFST>0THENSD=SD+ST
2510 PRINT"NET PROFIT TO BE PAID TO INVESTORS ";:PRINTUSINGP$;SD
2520 PRINT"LEFT FROM FRONT MONEY";USINGP$;FM
2540 IFFM<0GOTO2780
2560 GOSUB2850
2580 GOTO20?0
2600 DATA"DIRECTOR",15000,500,7500,250,2500,100,10,1,.25
2610 DATA"MALE STAR",0,20000,0,10000,0,2500,10,1,.5
2620 DATA"FEMALE STAR",0,16000,0,8000,0,2000,7,1,.75
2630 DATA"SET DESIGNER",10000,300,5000,200,1500,50,3,1,.85
2640 DATA"COSTUME DESIGNER",10000,300,5000,200,1500,50,2,1,.9
2650 DATA"LIGHTING DESIGNER",8000,300,4000,200,1000,50,1.5,1,.8
2660 DATA"SOUND DESIGNER",4000,200,2000,100,750,25,2,1,.75
2670 DATA"CHOREOGRAPHER",10000,300,6000,150,2000,75,3,1,.7
2680 DATA"COMPOSER",20000,800,10000,400,5000,150,8,1,.7
2690 DATA"LYRICIST",15000,600,7500,300,3000,150,7,1,.8
2700 DATA"BOOK AUTHOR",15000,600,7500,300,3000,150,7,1,.8
2710 DATA"ARRANGER",8000,600,5000,400,1500,100,4,1,.85
2720 DATA"SETS",150000,200,75000,100,50000,75,2,1,.75
2730 DATA"LIGHTS",3000,750,1500,500,1000,200,2,1,.75
2740 DATA"COSTUMES",100000,2000,50000,1000,15000,250,2,1,.75
2750 DATA"SOUND",3000,1000,1500,500,1000,200,2,1,.75
2760 DATA"PROPS",40000,1000,20000,500,5000,150,1.75,1,.9
2770 'YOU'VE SPENT MORE MONEY THAN YOU HAVE AVAILABLE
2780 CLS:PRINT@192,"YOU HAVE SPENT MORE MONEY THAN YOU HAVE.":PR
INT:PRINT"THE STATE ATTORNEY GENERAL'S OFFICE WILL CONTACT YOU B
Y THE":PRINT
2785 PRINT"FIRST OF THE MONTH. THEY ASK THAT BEFORE THEN YOU TAK
E THE TIME":PRINT:PRINT"TO SEE 'THE PRODUCERS' WITH ZERO MOSTEL
TO SEE WHERE YOU WENT":PRINT:PRINT"WRONG."
2800 PRINT:PRINT:INPUT"SHOW-BIZ STILL IN YOUR BLOOD";TA$:IFLEFT$
(TA$,1)="Y"RUN
2810 CLS:END
2830 R=RND(10):R1=R#IV:R1=INT(R1/9):FORTI=1TO1500:NEXT:RETURN
2850 PRINT:INPUT"DO YOU WANT TO CLOSE THE SHOW";C9$
2860 IFLEFT$(C9$,1)="Y"THEN3000ELSERETURN
```

```
2880 PRINT:INPUT"HOW MUCH DO YOU WANT TO SPENT ON ADVERTISING TH
IS WEEK (DO NOT  USE DOLLAR SIGN OR COMMAS)";AD:RETURN
2900 PRINT"THE CURRENT TICKET PRICE IS";USINGP$;SP:INPUT"DO YOU
WANT TO CHANGE THE TICKET PRICE";YN$
2910 IFLEFT$(YN$,1)="N"THENRETURN
2920 INPUT"TO WHAT PRICE";SN
2930 IFSN=SPPRINT"THAT'S THE CURRENT TICKET PRICE":GOTO2900
2940 IFSN>SPTHEND=(SN-SP)*2:P9=P9-D
2950 IFSN<SPTHEND=(SP-SN)*2:P9=P9+D
2960 SP=SN:RETURN
2980 SE=SD*.9:SD=SD-SE:SF=SF+SE:PRINT"90% OF PROFITS DISTRIBUTED
 TO INVESTORS":FORB=1TO1500:NEXT:RETURN
2990 'CLOSING NOTICE
3000 CLS:FORX=0TO127:SET(X,0):SET(X,47):NEXT
3010 FORY=0TO47:SET(0,Y):SET(127,Y):NEXT
3020 PRINT@86,PR$;" PRODUCTIONS INC.";
3030 PRINT@196,"C L O S I N G  N O T I C E";
3040 PRINT@324,"IT IS WITH GREAT REGRET THAT WE POST THIS NOTICE
.";
3050 PRINT@388,"WE WANT TO THANK ALL OF YOU FOR ALL OF YOUR HARD
";
3060 PRINT@452,"WORK. PLEASE BE ADVISED THAT THIS PRODUCTION WIL
L";
3070 PRINT@516,"CLOSE TWO WEEKS FROM TONIGHT. AGAIN THANK YOU.";
3080 PRINT@670,"SINCERELY YOURS,";
3090 PRINT@734,PR$;" PRODUCTIONS";
3100 PRINT@836,"P.S. YOUR FINAL TOTALS TO FOLLOW";
3110 FORTI=1TO7000:NEXT
3120 'FINAL TOTALS
3130 CLS:PRINT"FINAL TOTALS:":PRINT:PRINT"YOUR SHOW RAN FOR";WK;
"WEEKS BEFORE CLOSING.":PRINT"THAT'S";WK*8;"PERFORMANCES.":PRINT
"OUT OF YOUR ORIGINAL $1,000,000 YOU HAD";
3140 PRINTUSINGP$;FM;:PRINT" LEFT WHICH    HAS BEEN GIVEN BACK T
O YOUR INVESTORS."
3150 PRINT"TOTAL AMOUNT PAID TO INVESTORS":PRINTUSINGP$;SD+SF+FM
3160 PRINT"THAT MAKES A ";((SD+SF+FM)-1000000)/10000;"% RETURN O
N THEIR INVESTMENT."
3170 PRINT"(A 0% RETURN IS THE BREAK-EVEN POINT FOR YOUR INVESTO
RS":PRINT"A NEGATIVE RETURN IS A LOSS)":PRINT:PRINT"DO YOU WANT
TO PRODUCE ANOTHER SHOW";:INPUTYN$:IFLEFT$(YN$,1)="Y"THENRUNELSE
END
3180 'ERROR TRAPING ROUTINE
3190 PRINT(ERR/2)+1;"ERROR IN LINE";ERL:FORTI=1TO2500:NEXT:RESUM
ENEXT
```

TRS-80® SWAT TABLE FOR:
BROADWAY

LINES	SWAT CODE	LENGTH
10 - 170	FL	359
180 - 360	HP	556
380 - 450	HQ	550
460 - 520	KZ	522
530 - 660	IX	421
670 - 750	ZY	535
755 - 780	AK	652
810 - 850	GX	562
860 - 1000	IE	311
1010 - 1090	VG	601
1100 - 1210	FK	523
1220 - 1340	IQ	644
1360 - 1430	LH	526
1440 - 1550	WI	522
1560 - 1700	DS	412
1710 - 1810	JE	568
1820 - 1890	TE	502
1900 - 1970	EP	535
1980 - 2055	TT	513
2060 - 2230	SU	404
2240 - 2330	MT	551
2340 - 2440	SL	507
2450 - 2600	CH	329
2610 - 2700	LL	537
2710 - 2785	KZ	636
2800 - 2940	ND	501
2950 - 3070	FF	537
3080 - 3170	VQ	654
3180 - 3190	YF	76

NOTES:

LEYTE

by Victor A. Vernon Jr. TRS-80 version by Alan J. Zett

***Leyte* is a sea-battle simulation for an TRS-80 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 Bernar-

dino 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.

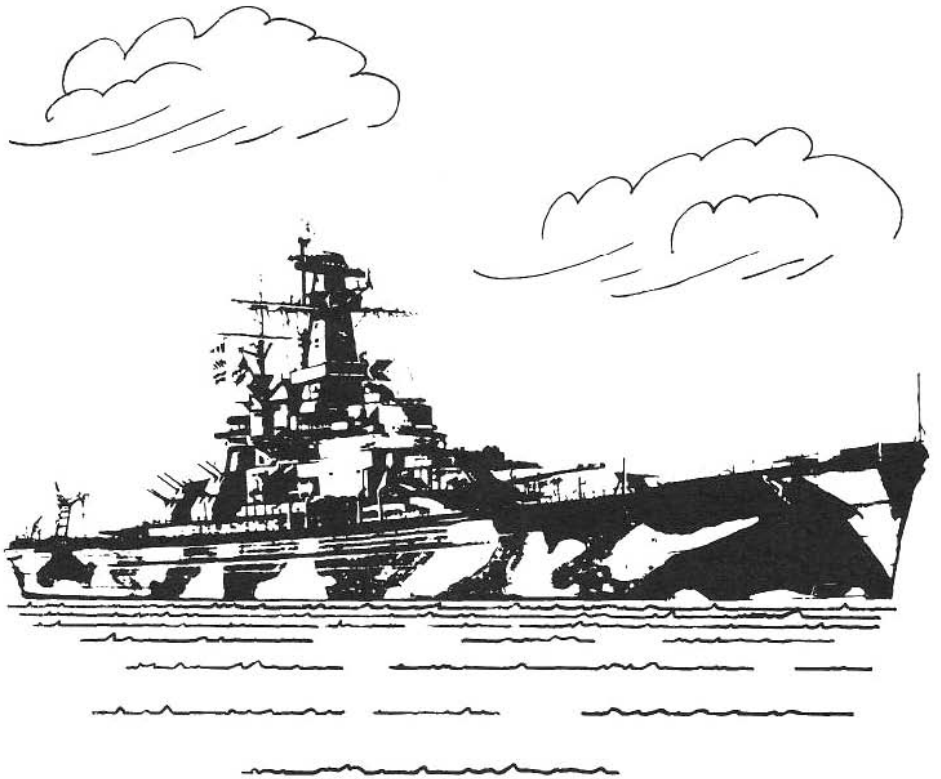
About this time, Halsey ordered the formation of a new task group, under Admiral Lee, whose job it was to guard the San Bernardino Strait if and when Halsey found the Japanese carriers. Meanwhile, The Japanese Northern force was trying desperately to be found, but was having little success. Although 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, you

lose. 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        TRS-80 BASIC         SS
SS           'Leyte'           SS
SS Author:  Victor Vernon Jr.  SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
50 POKE 16553,255
```

Initialization.

```
60 CLEAR200:DIMA(7,7),J(7,7),A$(7),J$(7),H(4,6,1):RANDOM
65 ON ERROR GOTO7000
70 CLS:PRINT"ONE MOMENT PLEASE":PRINT"THERE IS A LOT OF DATA TO
READ":NT=50:JT=100:A=265:A1=-1:I=15360
```

Load the arrays.

```
100 FORX=0TO7:READJ$(X),A$(X):FORY=0TO7:READJ(X,Y),A(X,Y):NEXTY,
X
110 FORX=0TO4:FORY=1TO6:FORZ=0TO1:READH(X,Y,Z):NEXTZ,Y,X
```

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

```
115 FORX=0TO640STEP64:PRINT@X,STRING$(64,46):NEXTX
120 PRINT@394,STRING$(39,191);:PRINT@454,STRING$(49,191);:PRINT@
513,STRING$(56,191);:PRINT@576,STRING$(62,191);:PRINT@578,"< SAN
 BERNARDINO STR.";
130 PRINT@620,"LEYTE GULF >";:POKE15821,83:POKE15828,65:POKE1583
5,77:POKE15842,65:POKE15849,82
135 FORX=15822TO15850STEP7:POKEX,32:POKEX-2,32:NEXTX
```

Move the fleets.

```
150 FORX=1TO1:NEXTX:C=0:H=0
160 PRINT@640,CHR$(31):POKEI+J,46:POKEI+A,46
165 READJ:IFJ=999THEN5010
170 M1=3:Z=RND(3):IFZ=1THENJ=J+64ELSEIFZ=2THENJ=J-64
175 IFPEEK(I+A+A1)=191PRINT@640,"YOU'RE ABOUT TO RUN AGROUND TUR
KEY":PRINT"THE COMPUTER WILL NOW CORRECT YOUR STUPIDITY":FORZZ=1
TO500:NEXTZZ:A1=-64
176 IFA+A1<0THENA1=64
180 A=A+A1:POKEI+A,65:POKEI+J,74:IFA=JTHENPOKEI+A,42
```

Command routine.

```
190 PRINT@640,"COMMAND ?";CHR$(220):IFJ=A+1ORJ=A-1ORJ=A+64ORJ=A-
64THENM1=0
200 PRINT" 1 - STATUS REPORT":IFA=JTHENM1=-1
210 PRINT" 2 - FIRE GUNS":IFA=J+63ORA=J-63ORA=J+65ORA=J-65THENM1
=2
220 PRINT" 3 - FIRE TORPEDOES":IFA=J+1ORA=J-1ORA=J+64ORA=J-64THE
NM1=0
230 PRINT" 4 - COURSE CORRECTION":M=M1
240 FORX=1TO2000:I$=INKEY$:IFI$=""THEN270
250 IFASC(I$)<49ORASC(I$)>52THEN270ELSEFORX=1TO1:NEXTX
260 G=VAL(I$):ONG GOTO280,480,620,730
270 NEXTX:C=C+1:IFC>2THEN150ELSE240
```

Display the status of the American ships.

```
280 FORX=0TO7:IFA(X,5)<1THENR$="SUNK":GOTO430
290 IFX>1THEN340
300 IFA(X,4)<500RA(X,2)<7THENR$="FLOATING JUNK YARD":GOTO430
310 IFA(X,4)<1000RA(X,2)<12THENR$="VERY HEAVY DAMAGE":GOTO430
320 IFA(X,4)<1500RA(X,2)<17THENR$="MODERATE DAMAGE":GOTO430
:FORZZ=1TO500:NEXTZZ:H=0
330 R$="ESSENTIALLY UNDAMAGED":GOTO430
340 IFX>4THEN380
```

```
350 IFA(X,4)<500RA(X,2)<7THENR$="PREPARE TO ABANDON SHIP!":GOTO4
30
360 IFA(X,4)<1000RA(X,2)<14THENR$="SEVERE DAMAGE":GOTO430
370 R$="LITTLE OR NO DAMAGE":GOTO430
380 IFX=7THEN420
390 IFA(X,4)<200RA(X,2)<4THENR$="SINKING!":GOTO430
400 IFA(X,4)<500RA(X,2)<8THENR$="HEAVY DAMAGE!":GOTO430
410 R$="LIGHT OR NO DAMAGE":GOTO430
420 IFA(X,4)<250RA(X,2)<5THENR$="HEAVY DAMAGE!"ELSER$="UNDAMAGED
"
430 PRINT@640,CHR$(31);"STATUS :"
440 PRINTA$(X);" : ";R$:IFX=7 ANDA(7,5)>1PRINTA(7,5)" AFLOAT"
460 FORY=1TO500:NEXTY:NEXTX
470 GOTO190
```

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

```
480 C=C+1:FORX=0TO7:IFA(X,5)<1THENNEXTX
```

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

```
490 PRINT@640,CHR$(31);"ENTER TARGET FOR "A$(X):M=M1
500 T$=INKEY$:IFT$=""THEN500ELSET=VAL(T$)
510 IFT<1ORT>8PRINT@704,"ENTER NUMBER OF TARGET SHIP. CHECK INST
RUCTIONS":GOTO500
```

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

```
520 T=T-1:M=M+J(T,7):J(T,7)=J(T,7)+.5
```

If target is sunk, select another.

```
530 IFJ(T,5)<1PRINTJ$(T)" SUNK":FORZZ=1TO200:NEXTZZ:GOTO490
```

Reset hit counter to zero.

```
540 H=0:FORY=1TOA(X,6):Z=RND(10)+M:IFZ>4THEN570ELSEH=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=RND(5)-1:IFX<5AND(V=0ORV=1ORV=3)THENJ(T,6)=J(T,6)-1
```

Timing delay.

```
555 FORZ=1TO200:NEXTZ
```

Change defense factors on target.

```
560 J(T,2)=J(T,2)-H(V,A(X,0),1):J(T,4)=J(T,4)-H(V,A(X,0),0)
```

Display the number of main armament hits.

```
570 NEXTY:IFH>0PRINT@704,H"MAIN ARMAMENT HITS ON "J$(T);CHR$(31)
```

Repeat the process for secondary armaments.

```
580 M=M1:FORY=1TOA(X,6)*2:Z=RND(10)+M:IFZ>4THEN610ELSEH=H+1
600 V=RND(5)-1:J(T,2)=J(T,2)-H(V,A(X,1),1):J(T,4)=J(T,4)-H(V,A(X
,1),0)
610 NEXTY,X:M=M1:IFH>0PRINT@768,H"SECONDARY ARMAMENT HITS ON "J$
(T);CHR$(31):H=0
615 FORZZ=1TO500:NEXTZZ:GOTO880
```

U.S. torpedo fire.

```
620 C=C+1:M=M1-C:PRINT@640,CHR$(31)
630 IFA(7,3)<1ANDNT<1PRINT"NO TORPEDOES TO FIRE":FORZZ=1TO500:NE
XTZZ:GOTO880
640 IFA(7,3)<1ANDNT>9THENA(7,3)=10:NT=NT-10
650 A(7,3)=A(7,3)-5:PRINT"TARGET ? (ENTER NUMBER PLEASE)":M=M1
670 T$=INKEY$:IFT$=""THEN670ELSET=VAL(T$):IFT<1ORT>8PRINT"NO SUC
H TARGET":GOTO670
680 T=T-1:IFJ(T,5)<1PRINT@704,J$(T)" SUNK SELECT ANOTHER TARGET"
:GOTO670
690 FORY=1TO5:Z=RND(10)+M:IFZ>5THEN720
700 V=RND(5)-1:J(T,2)=J(T,2)-H(V,6,1):J(T,4)=J(T,4)-H(V,6,0):H=H
+1
710 PRINT@704,H"HITS ON "J$(T);CHR$(220):FORZZ=1TO200:NEXTZZ
720 NEXTY:GOTO880
```

U.S. course change routine.

```
730 PRINT@640,CHR$(31);"   8  1  2":PRINT@710,"!":PRINT@771,"7--
+--3   ENTER NEW COURSE":PRINT@838,"!":PRINT@899,"6  5  4";
760 I$=INKEY$:IFI$=""THEN760ELSEG=VAL(I$)
770 IFG<1ORG>8THEN760
```

```
780 ONGGOTO790,800,810,820,830,840,850,860
790 A1=-64:GOTO870
800 A1=-63:GOTO870
810 A1=1:GOTO870
820 A1=65:GOTO870
830 A1=64:GOTO870
840 A1=63:GOTO870
850 A1=-1:GOTO870
860 A1=-65
870 GOTO190
```

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

```
880 IFC>1ANDM1<3THEN1020ELSEM=M1:PRINT@640,CHR$(31);:H=0
```

Japanese gunfire routine.

```
890 FORY=0TO7:IFJ(Y,5)<1THEN1010
895 PRINT@640,"INCOMING JAPANESE FIRE FROM ";J$(Y);CHR$(31)
900 FORX=0TO7:IFA(X,5)<1THENNEXTX:GOTO4070
910 T=TT:FORX=1TO1:NEXTX:TT=TT+1:IFTT>7THENTT=0
915 IFA(T,5)<1THEN910
920 FORX=1TOJ(Y,6):Z=RND(10)+M:IFZ>5THEN960
940 M=M+.5:H=H+1:V=RND(5)-1:IF(V=1ORV=3)ANDY<4THENA(T,6)=A(T,6)-
1
950 A(T,2)=A(T,2)-H(V,J(Y,0),1):A(T,4)=A(T,4)-H(V,J(Y,0),0)
960 NEXTX:IFH>0PRINT@768,H"MAIN ARMAMENT HITS ON "A$(T);CHR$(31)
:FORZZ=1TO500:NEXTZZ:H=0
965 IFY>5THEN1010
970 FORX=1TOJ(Y,7)$2:Z=RND(10)+M:IFZ>5THEN1000
980 H=H+1:V=RND(5)-1:A(T,2)=A(T,2)-H(V,J(Y,1),1):A(T,4)=A(T,4)-H
(V,J(Y,1),0)
1000 NEXTX:IFH>0THENPRINT@768,H"SECONDARY ARMAMENT HITS ON "A$(T
):H=0:FORZZ=1TO500:NEXTZZ
1010 NEXTY:FORX=1TO100:NEXTX
1015 IFC>1THEN3020ELSE160
```

Japanese torpedo fire routine.

```
1020 M=M1:PRINT@640,CHR$(31);"INCOMING JAPANESE TORPEDOE FIRE":I
FM1=3THENM=M+1
1025 FORZZ=1TO300:NEXTZZ
1030 FORX=4TO6:IFJ(X,3)>0ANDJ(X,5)>0THEN2040ELSENEXTX
1040 IF(J(7,3)<1ANDJT(1)ORJ(7,5)<1THEN3020
```

```
1050 IFJ(7,3)<1ANDJT>9THENJ(7,3)=10:JT=JT-10
1060 J(7,3)=J(7,3)-5
1070 FORX=0TO7:IFA(X,5)<1THENNEXTX:GOTO4070
1080 T=TT:FORX=1TO1:NEXTX:TT=TT+1:IFTT>7THENTT=0
1085 IFA(T,5)<1THEN1080
1090 FORX=1TO5:Z=RND(10)+M:IFZ>4THEN2030
1100 V=RND(5)-1:A(T,2)=A(T,2)-H(V,6,1):A(T,4)=A(T,4)-H(V,6,0):PR
INT@704,"TORPEDOE HIT ON "A$(T);CHR$(31):FORZZ=1TO500:NEXTZZ:IFT
<7THENT=T+1
1110 FORZZ=1TO500:NEXTZZ
2030 NEXTX:GOTO3020
2040 F=X:FORX=1TO1:NEXTX:J(F,3)=J(F,3)-8
2050 FORX=0TO7:IFA(X,5)<1THENNEXTX:GOTO4070
2060 T=X:FORX=1TO1:NEXTX:IFT=8THEN4070
2070 FORX=1TO8:Z=RND(10)+M:IFZ>4THEN3010
2080 V=RND(5)-1:A(T,2)=A(T,2)-H(V,6,1):A(T,4)=A(T,4)-H(V,6,0):PR
INT@704,"TORPEDOE HIT ON "A$(T);CHR$(31):FORZ=1TO500:NEXTZ:IFT<7
THENT=T+1
3010 NEXTX
```

Status update.

```
3020 PRINT@640,CHR$(31);"CHECKING STATUS OF ALL SHIPS":FORZZ=1TO
400:NEXTZZ
3030 FORX=0TO6:J(X,7)=0:IFJ(X,2)<1ORJ(X,4)<1THENJ(X,5)=0:PRINTJ$
(X)" SUNK",:FORZ=1TO500:NEXTZ
3050 IFA(X,2)<1ORA(X,4)<1THENA(X,5)=0
3060 NEXTX:J(7,7)=0:IFA(7,2)>0ORA(7,4)>0THEN3090
3070 A(7,5)=A(7,5)-1:IFA(7,5)<1THEN4020
3080 A(7,2)=5:A(7,4)=20:A(7,3)=10:NT=NT-10
3090 IFJ(7,2)>0ORJ(7,4)>0THEN4020
4000 J(7,5)=J(7,5)-1:IFJ(7,5)<1THEN4020
4010 J(7,2)=5:J(7,4)=25:J(7,3)=10:JT=JT-10
4020 FORX=0TO6::IFJ(X,5)<1THENNEXTXELSEGOTO4060
```

End-of-game messages.

```
4030 CLS:PRINTCHR$(23):PRINT"ALL JAPANESE CAPITOL SHIPS SUNK":PR
INT
4040 PRINT"YOU HAVE SAVED THE LEYTE":PRINT:PRINT"LANDINGS":PRINT
:PRINT
4050 PRINT"TO PLAY AGAIN TYPE RUN":END
4060 FORX=0TO7:IFA(X,5)>0THEN150
4070 NEXTX:CLS:PRINTCHR$(23):PRINT"YOU LOST ALL YOUR SHIPS":PRIN
T
```

```
4080 PRINT"MAYBY KINKAID'S SEVENTH FLEET":PRINT
4090 PRINT"WILL SAVE THE LEYTE LANDINGS":PRINT:PRINT:GOTO4050
5010 CLS:PRINTCHR$(23):PRINT"THE JAPANESE HAVE REACHED LEYTE":PR
INT:GOTO4080
```

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. DESTROYERS",5,5,0,5,10,15,10,1
0,50,55,8,6,6,5,0,0
```

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 384,385,386,387,388,325,326,327,264,265,266,267,268,26
9,270,207,208,273,274,211,212,213,214,215,216,217,218,283,284,22
1
6120 DATA 286,287,224,289,290,291,228,229,230,295,296,233,298,29
9,300,301,301,303,304,305,306,371,372,439,440,441,506,507,508,50
9
6130 DATA 510,574,999,999
```

Erase to bottom of screen.

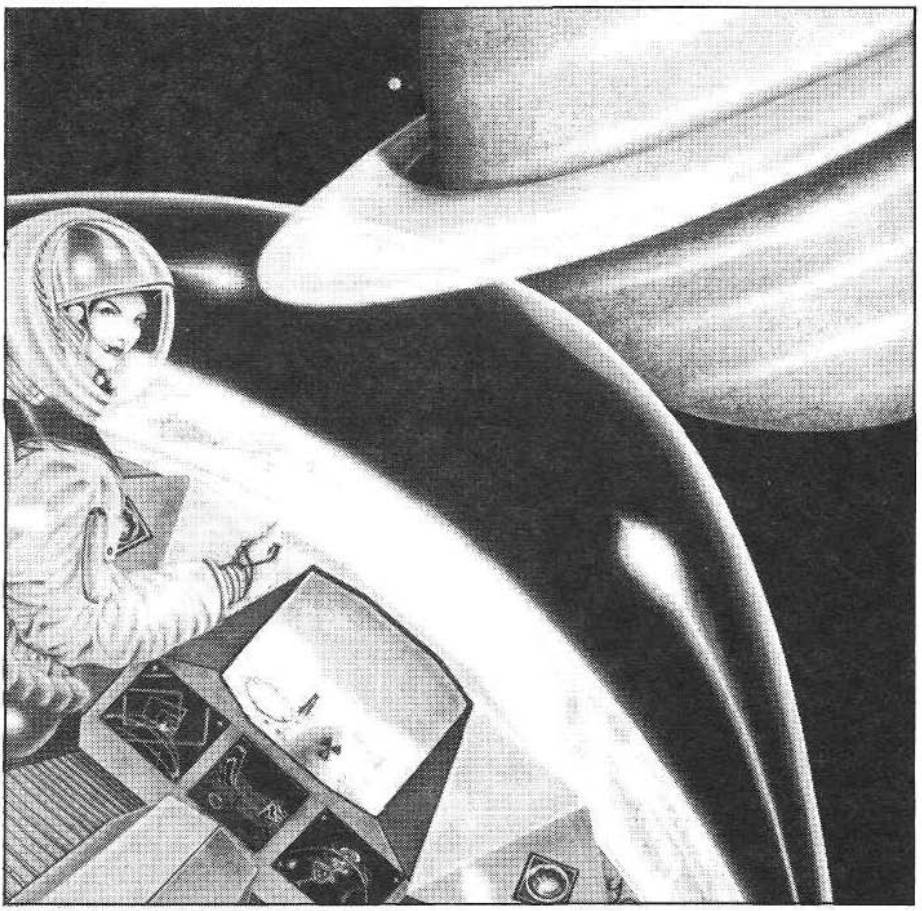
```
7000 FORX=1TO1:NEXTX:FORY=1TO1:NEXTY:FORZZ=1TO1:NEXTZZ
7100 RESUME150
```

TRS-80® SWAT TABLE FOR:
LEYTE

LINES	SWAT CODE	LENGTH
50 - 135	YA	513
150 - 220	FP	518
230 - 340	FU	419
350 - 470	XC	445
480 - 570	DS	504
580 - 680	TB	523
690 - 820	TW	402
830 - 920	UT	307
940 - 1025	OX	511
1030 - 2040	YJ	435
2050 - 3090	WD	527
4000 - 6000	SQ	518
6010 - 6080	AB	534
6090 - 7100	MG	416

NOTES:

TITAN



by William Morris and John Cope

***Titan* is an outer space mining simulation for a 16K TRS-80.**

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 Dilithium 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 in-

vestment 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 TRS-80 version, this was not possible, so a joystick emulator was set up. The arrow keys correspond to pushing a joystick in the indicated direction, and the ENTER key represents the joystick's fire button. Thus any reference to a joystick is easily understood by the TRS-80 user.

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.

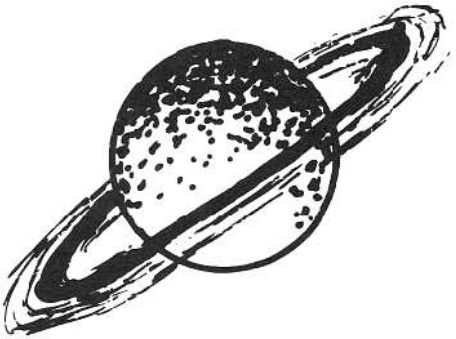
AC: Action flag indicating decision to buy or sell.
C: Cost of items purchased during Equipment phase, or the price of worker-related decisions.
C2: Choice flag which is set during the Labor Management phase.
CH: Choice variable.
DA: Month of the year.
EF: Efficiency rating.
EN: Energy consumption.
EQ$: Equipment.
EX$: Extras.
FF: Efficiency counter.
HO: Drill hole(s).
I$: Keyboard scan.
JB$: Packed graphic string of blanks to erase the robominer and drill rig.
JC$: Temporary graphics string.
JD$: Packed graphic string of drill rig.
JF$: Reverse screen Machine Language routine.
JM$: Sound routine string.
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        TRS-80 BASIC         SS
SS           'Titan'           SS
SS Authors: Wm Morris/J. 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 and title display.

```
1 REM ### TITAN ###
2 REM ### (C) WM. MORRIS & J. COPE ###
9 REM ### INITIALIZATION ###
10 CLS:CLEAR300:DEFSTRJ:DEFINTW-Z:DIM C(12),K(4,16):GOSUB20000:G
OSUB30000:GOSUB50000
19 REM ### TITLE ###
20 CLS:GOSUB20010:J=STRING$(10,32)+STRING$(43,128)+STRING$(11,32
):PRINT@192,J;J;J;J;J;J;:FORZ=1TO5:FORY=1TO18:READX:PRINT@X+268,
CHR$(191);:NEXT:READUN,UL:GOSUB20100:NEXT
30 PRINT@657,"(C) WM. MORRIS & J. COPE 1981"
40 FORZ=1TO4:GOSUB30010:US=USR(0):UN=162:UL=20:GOSUB20010:GOSUB2
0100:GOSUB30010:US=USR(0):UN=217:UL=18:GOSUB20010:GOSUB20100:NEX
T:FORZ=1TO1000:NEXT:GOSUB9000
```



Playing parameters.

```
99 REM *** PLAYING PARAMETERS ***
100 GOSUB20110:CLS:PRINTCHR$(23);:PRINT@144,"WELCOME TO TITAN!"
110 PRINT@338,"USE THE ARROWS TO SELECT THE        NUMBER OF PLA
YERS.":GOSUB200
120 PL=X:CLS:GOSUB20110:PRINTCHR$(23):PRINT@396,"LEVEL OF DIFFIC
ULTY?":GOSUB 200
130 LV=X+4:CLS:GOSUB20110:PRINTCHR$(23)
140 FOR Z=1 TO PL:PRINT@-128+192*Z,"   PLAYER # ";Z;" COMMANDS B
ASE           ";MID$(A$,Z*10-9,10):NEXT
150 PRINT@906,"PRESS ENTER TO BEGIN."
160 GOSUB40000:IFI<>13THEN160
170 FOR Z=1 TO PL:K(Z,6)=1:K(Z,13)=500:K(Z,14)=100:K(Z,15)=5:NEX
T Z:GOTO 300
199 REM *** JOYSTICK SELECTION ***
200 PRINT@848,"";:Y=1
210 PRINT"*1*  2   3   4 ":GOTO250
220 PRINT" 1  *2*  3   4 ":GOTO250
230 PRINT" 1   2  *3*  4 ":GOTO250
240 PRINT" 1   2   3  *4*":GOTO250
250 PRINT@848,"";:GOSUB40000:IFI=13 THEN X=Y:RETURN
260 IF I <> 8 AND I <> 9 THEN 250
270 IF I=8THEN Y=Y-2:IF Y<0 THEN Y=0
280 Y=Y+1:IF Y>4 THEN Y=1
290 ON Y GOTO 210,220,230,240
```

Game play sequence.

```
299 REM *** GAME SEQUENCE ***
```

```
300 GOSUB 7000:GOSUB 8000:FOR P=1 TO PL:GOSUB 1000:NEXT P:GOSUB
6000:GOTO 300
```

Equipment-management phase.

```
999 REM *** DISPLAY ***
1000 CLS:PRINTCHR$(23);:CH=1:AC=0:C2=0
1010 PRINT@16,"BASE:";MID$(A$,P*10-9,10):PRINT@128,DA;"/1/";YE,"
 STATUS ";K(P,0)
1020 PRINT@256,"EQUIPMENT     OWNED     COST":PRINT:FOR Z=1 TO 6
:PRINT MID$(EQ$,Z*10-9,10):NEXT Z
1030 FOR Z=1 TO 6:PRINT@348+Z*64,K(P,Z);" ";:PRINT@366+Z*64,C(Z)
;:NEXT Z
1040 PRINT@848,"CREDIT  ";K(P,13);"  ":GOSUB 1100:IF CH=7 THEN C
H=1:AC=0:GOTO 1300
1050 GOTO 1020
```

Buy-and-sell routine. Accessed by equipment-management and labor-relations phases.

```
1099 REM *** BUY & SELL ***
1100 IF CH>7 THEN CH=CH-7
1110 IF AC=0 THEN PRINT@960,"INCREASE";:PRINT@1004,"         ";
1120 IF AC=1 THEN PRINT@1004,"DECREASE";:PRINT@960,"         ";
1125 IF C2=0 THEN PRINT@982,MID$(EQ$,CH*10-9,10);:GOTO 1140
1130 PRINT@982,MID$(WC$,CH*10-9,10);
1140 GOSUB40000
1150 IF I=13 THEN 1200
1160 IF I=8 THEN AC=0
1170 IF I=9 THEN AC=1
1180 IF I=10 THEN CH=CH+1
1190 IF I=91 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) OR (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 THEN 1250
1230 IF AC=0 AND K(P,13)<C(CH) THEN 1280
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,14)=K(P,14)+1
1260 IF C2 THEN CH=CH+1:IF CH=8 THEN RETURN
1270 GOSUB20120:RETURN
```

```
1280 IF C2 THEN CH=CH+1
1290 UN=255:UL=5:GOSUB20100:PRINTCHR$(23);:RETURN
```

Labor-relations phase.

```
1299 REM *** LABOR RELATIONS ***
1300 CLS:PRINTCHR$(23);:C2=1
1310 PRINT@16,"BASE:";MID$(A$,P*10-9,10):PRINT@128,"MANPOWER";K(
P,14);
1320 K(P,7)=K(P,14)-(K(P,1)*LV)-(K(P,2)*LV*2)-(K(P,3)*LV*3)-(K(P
,4)*LV*4)-(K(P,5)*LV*5)-(K(P,6)*LV*10)
1330 C(9)=K(P,14):C(10)=K(P,14)*2:C(11)=K(P,14)*3:C(12)=K(P,14)*
4:EN=K(P,6)*500*(11-LV)
1340 GOSUB 5000:PRINT@158,"EFFICIENCY";K(P,15);"%";
1350 PRINT@256,"CONDITIONS     SET      COST":PRINT:FOR Z=1 TO 6
:PRINT MID$(WC$,Z*10-9,10):NEXT Z
1360 FOR Z=1 TO 6:PRINT@348+Z*64,K(P,Z+6);" ";:PRINT@366+Z*64,C(
Z+6);:NEXT Z
1370 PRINT@848,"CREDIT  ";K(P,13);"  ":GOSUB 1100:IF CH=7 THEN 2
000
1380 GOTO1340
```

Mining phase and graphics display.

```
1999 REM *** MINING PHASE ***
2000 IF K(P,1)<1 THEN 3000
2010 CLS:PRINT@192,STRING$(64,131):PRINT@832,STRING$(64,176)
2030 FORY=0TO2:FORZ=0TO4:R=INT(RND(127)):SET(R,Y):NEXTZ:NEXTY
```

Establish the location of the Dilithium 3 crystals.

```
2040 V1=INT(RND(109)+9):V2=INT(RND(31)+9):RA=9-LV:RB=INT(RND(2)-
1)
2100 HO=0
2120 IF K(P,1)=0 AND (HO=0 OR K(P,2)=0) THEN 2720
2130 PRINT@960,"                ";:PRINT@1007,"DRILL RIG(S)";:X=
1:W=124:UN=81:UL=10:JC=JD:GOTO 2150
2140 PRINT@1007,"               ";:PRINT@960,"ROBOMINER(S)";:X=2
:W=65:UN=108:JC=JR
2150 PRINT K(P,X);:PRINT@W,JC;:GOSUB20100
2160 GOSUB40000:IFI=13THEN2200
```

Inkey input triggers the appropriate screen display.

```
2165 IF I=9 THEN PRINT@W,JB;:GOTO2130
```

```
2170 IF I=8 THEN PRINT@W,JB;:GOTO2140
2175 IF I=91 THEN 2185
2180 GOTO2160
2185 PRINT@960,STRING$(63,32);:PRINT@988,"CONTINUE";:PRINT@W,JB;
:UN=40:GOSUB20100
2190 GOSUB40000:IFI=13THEN3000 ELSE IF I<8ORI>9 THEN2185
2195 PRINT@988,"        ";:GOTO 2165
2200 IF K(P,X)<1 OR (X=2 AND HO=0) THEN GOSUB20140:GOTO2160
2210 K(P,X)=K(P,X)-1:UN=40:GOSUB20100
2220 I$=INKEY$:IFI$=""THEN2230 ELSEI=ASC(I$):IFI=13THEN 2300 ELS
E IFI>7ANDI<10THENPRINT@W,JB;
```

Horizontal movement phase.

```
2230 PRINT@W,JB;:IFI=8THENW=W-1:IFW<66ANDX=1THENW=66
2235 IF W<65 THEN W=65
2240 IFI=9THENW=W+1:IFW>124THENW=124
2250 PRINT@W,JC;:GOTO2220
2300 IF X=1 AND (W<66 OR W>124) THEN 2220
2310 IF X=2 THEN 2400
2320 W=(W-63)*2+1:Y=9
```

Vertical drilling sequence.

```
2330 SET(W,Y):VE=2
2333 IF I=10 THEN Y=Y+1:OUT255,0:OUT255,1:VE=4:IF Y>40THENY=40
2335 GOSUB 2500:PRINT@1008,"ENERGY";EN;:IF EN<1 OR VE=1 THEN 260
0
2340 OUT255,0:OUT255,6:I$=INKEY$:IFI$=""THEN2330ELSEI=ASC(I$):IF
I<>13THEN2330
2360 PRINT@(W-1)/2+63,JB;:HO=HO+1:GOTO 2120
```

Robominer vertical drilling sequence. Allows up and down movement.

```
2400 W1=W:W=(W-63)*2+3:Y=10:IFPOINT(W,Y)=0THENW=(W-3)/2+63:GOTO2
220
2410 Y=8
2420 SET(W,Y):I$=INKEY$:IFI$=""THEN2425 ELSE I=ASC(I$)
2425 IFI=10THENY=Y+1:IFPOINT(W,Y)=0THENY=Y-1
2430 IF I=91 THEN SET(W,Y):Y=Y-1:IF Y<8 THEN Y=8
2435 VE=2:GOSUB 2500:OUT255,0:OUT255,1:PRINT@1008,"ENERGY";EN;:I
F EN<1 THEN 2600
2440 IF(I<>8ANDI<>9) OR Y<8THEN 2420
2445 SET(W,Y):I$=INKEY$:IFI$=""THEN2450 ELSE I=ASC(I$)
```

```
2450 VE=2:SET(W,Y):IFI=9 THEN W=W+1:OUT255,0:OUT255,1:VE=4:IF W>
126 THEN W=W-1
2460 IFI=8 THEN W=W-1:OUT255,0:OUT255,1:VE=4:IF W<1 THEN W=W+1
2465 GOSUB 2500:PRINT@1008,"ENERGY";EN;:IF EN<1 OR VE=1 THEN 260
0
2470 SET(W,Y):IFI<>13THEN2445
2480 PRINT@W1,JB;:GOTO2120
```

Compare distance from drill rig bit or robominer to Dilithium.

```
2500 RESET(W,Y):EN=INT(EN-(100/K(P,15))*VE)
2510 VE=0:PRINT@960,STRING$(63,32);:PRINT@960,"";:SET(W,Y):IF Y<
10 THEN PRINT "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) THENPRINT " 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) THENPRINT "THIRIDIUM     ";:RETURN
2540 IF (W<V1-RA OR W>V1+RA*2) OR (Y<V2-RB OR Y>V2+K(P,4)+RB*2)
THEN PRINT "SECONIUM      ";:RETURN
2550 IF (W<V1 OR W>V1+RA) OR (Y<V2 OR Y>V2+K(P,4)+RB) THEN PRINT
 'FIRIDIUM      ";:RETURN
2560 VE=1:RETURN
```

Discovery of Dilithium.

```
2600 IF EN<1 THEN 2800
2605 PRINT@960,STRING$(63,32);:PRINT@975,"EUREKA! YOU HAVE FOUND
 DILITHIUM!!!";
2610 FORZ=1TO4:GOSUB30010:U5=USR(0):GOSUB20010:GOSUB20110:NEXT
2620 PRINT@960,STRING$(63,32);:PRINT@1008,"STATUS";:X=0:Y=1:GOTO
 2640
2630 PRINT@960,STRING$(63,32);:PRINT@965,"500 CREDITS";:X=13:Y=5
00:GOTO 2640
2640 GOSUB40000:IFI=13 THEN K(P,X)=K(P,X)+Y:K(P,16)=K(P,16)+1:GO
TO 2700
2650 IFI=9THEN 2620
2660 IFI=8THEN 2630
2670 GOTO 2640
```

End of drilling operations due to lack of equipment or energy.

```
2700 IF EN<1 THEN 2800
2710 IF K(P,1)>0 THEN 2000
2720 PRINT@960,"YOU DO NOT HAVE ENOUGH EQUIPMENT LEFT TO CONTINU
E DRILLING.  ";:FORZ=1TO200:OUT255,0:OUT255,1:NEXT:GOTO3000
```

```
2800 PRINT@960,"YOU DO NOT HAVE ENOUGH ENERGY LEFT TO CONTINUE D
RILLING.     ";:FORZ=1TO200:OUT255,0:OUT255,1:NEXT
```

Cave-in sequence.

```
2999 REM ### ACCIDENTS ###
3000 R=RND(10)-K(P,11):IF R<7 OR K(P,16)<1 THEN 3500
3010 CLS:GOSUB20130:PRINTCHR$(23);:PRINT@22,"RED ALERT!"
3030 PRINT@394,"CAVE IN:";R=INT(RND(10))
3040 PRINT@528,R;"WORKERS LOST!":K(P,14)=K(P,14)-R
3050 IF R>6 THENPRINT@658,"SHAFT NOW CLOSED!":K(P,16)=K(P,16)-1
3060 GOSUB40000:IF I<>13 THEN 3060
3070 GOTO 4000
```

Meteor-storm sequence.

```
3499 REM ### METEOR STORMS ###
3500 R=RND(10)-K(P,3):IF R<8 THEN 4000
3510 CLS:GOSUB20130:PRINTCHR$(23);:PRINT@22,"RED ALERT!"
3530 PRINT@394,"METEOR STORM!:";R=INT(RND(10))
3540 PRINT@528,R;"WORKERS LOST!":K(P,14)=K(P,14)-R:IFR<7 THEN 35
90
3550 R=INT(RND(5)):IF K(P,R)<1 THEN 3590
3560 K(P,R)=K(P,R)-1:PRINT@658,MID$(EQ$,10*R-9,10);"DESTROYED!"
3590 GOSUB40000:IF I<>13 THEN 3590
```

Monthly report.

```
3999 REM ### 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+500+(50*K(P,0))+(Z*25*K(P,8
)))
4100 CLS:GOSUB20110:PRINT@0,"MONTHLY REPORT":PRINT@29,"TITAN":PR
INT@50,DA;"/1/";YE
4130 PRINT:PRINT"    ITEM          ACTAEON    BELLONA    CHIMERA
   DAEDALUS":PRINT
4140 FORZ=0TO6:W=Z*10:IFW=0 THEN PRINT "    STATUS    ";:GOTO417
0
4160 PRINT"    ";MID$(EQ$,W-9,10);
4170 FOR X=1 TO 4:PRINTTAB(7+X*11);K(X,Z);:NEXT:PRINT:NEXT
4180 FOR Z=1 TO 4:PRINT"    ";MID$(EX$,Z*10-9,10);
4190 FOR X=1 TO 4:PRINTTAB(7+X*11);K(X,Z+12);:NEXT:PRINT:NEXT
4210 PRINT@979,"PRESS ENTER TO CONTINUE";:GOSUB20130
4220 GOSUB40000:IF I<>13 THEN 4220
4230 RETURN
```

Efficiency rating algorithms.

```
4999 REM ### EFFICIENCY ###
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=FF/(K(P,6)*10)
5020 EF=EF*100:FF=K(P,7)*2.5:IF FF<0 THEN 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):IF 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.

```
5999 REM ### NEW COSTS ###
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)=4000
6080 NEXT Z:FOR Z=1 TO 4:FOR Y=8 TO 12:K(Z,Y)=0:NEXT Y:NEXT Z:RE
TURN
```

Date.

```
6999 REM ### NEW DATE ###
7000 DA=DA+1:IF DA=13 THEN DA=1:YE=YE+1
7010 RETURN
```

End-of-game sequence.

```
7999 REM ### END OF GAME ###
8000 IF YE=2051 AND DA>3 THEN 8500
8010 R=INT(RND(2)):IF R=1 OR YE=2050 THEN RETURN
8500 PRINT@978,"THE INSPECTOR HAS ARRIVED!";:GOSUB20140
8510 FORZ=1TO900:NEXT
8520 PRINT@974,"YOUR PERFORMANCE IS BEING ASSESSED.";:FOR Z=1 TO
 900:NEXT 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)
8540 NEXT X
```

```
8550 FOR X=1 TO PL:IF K(X,0)=Z THEN Y=Y+1
8560 NEXT X
8565 IFY=1THENGOTO8566ELSEGOTO8570
8566 FORX=1TOPL:IFK(X,0)=ZTHEN8590
8567 NEXTX
8570 FOR X=1 TO PL:IF K(X,0)=Z THEN GOSUB 8580
8571 NEXTX:GOTO8530
8580 K(X,0)=K(X,16)*100+K(X,15)+K(X,14)+K(X,13)+RND(9):RETURN
8590 PRINT@960,"THE SUPERVISOR OF";MID$(A$,(X*10)-9,10);" HAS WO
N THE RIGHT TO MINE TITAN!!";:GOTO8700
8600 IFK(1,0)>19 THEN 8630
8610 PRINT@960,"    YOU DID NOT ACHIEVE ENOUGH STATUS TO MINE AL
L OF TITAN!  ";:GOTO8700
8630 PRINT@960,"BECAUSE OF YOUR STATUS YOU HAVE WON THE RIGHT TO
 MINE TITAN!!";
8700 GOTO 8700
```

Initial values.

```
8999 REM *** INITIAL VALUES ***
9000 DA=0:YE=2050
9010 C(1)=100:C(2)=200:C(3)=300:C(4)=400:C(5)=500:C(6)=2000:C(7)
=5:C(8)=10:C(9)=100:C(10)=200:C(11)=300:C(12)=400
9020 EQ$="DRILL RIG ROBOMINER DEFLECTOR R&D UNIT  REFINERY  ENER
GIZER CONTINUE  "
9030 A$=" ACTAEON   BELLONA   CHIMERA   DAEDALUS  "
9040 WC$="LABOR POOLSHIFT TIMEWAGE SCALERECREATIONSAFETY    BONU
SES   CONTINUE  "
9050 EX$="CREDIT    MANPOWER  EFFICIENCY* VEINS * "
9060 RETURN
```

Data.

```
9999 REM *** DATA ***
10000 DATA 17,121,45,14,255,33,1,1,45,122,237,97,67,16,254,237,1
05,67,16,254,61,32,243,21,32,239,201
10010 DATA217,33,255,63,6,60,126,254,128,56,4,47,246,128,119,43,
124,184,48,242,217,201
10015 DATA175,155,160,24,24,24,26,183,183,139,160,143,144,24,24,
24,26,159,131,175,32,32,32,24,24,24,26,32,32,32
10020 DATA 1,2,3,4,5,6,67,68,131,132,195,196,1,1,1,1,1,1,162,30
10030 DATA 9,10,11,12,74,75,138,139,201,202,203,204,9,9,9,9,9,9,
108,30
10040 DATA 15,16,17,18,19,20,81,82,145,146,209,210,15,15,15,15,1
5,15,81,45
```

```
10050 DATA 24,25,26,27,87,88,91,92,151,152,153,154,155,156,214,2
15,220,221,64,20
10060 DATA 32,33,36,37,96,97,98,100,101,160,161,163,164,165,224,
225,228,229,68,50
```

Sound routines.

```
19999 REM *** SOUND ROUTINES ***
20000 JM="":FORZ=1TO27:READY:JM=JM+CHR$(Y):NEXT
20010 IFPEEK(16396)=201THEN20030 ELSE CMD"T":U=VARPTR(JM):U=PEEK
(U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536
20020 DEFUSR0=U:RETURN
20030 U=VARPTR(JM):POKE16526,PEEK(U+1):POKE16527,PEEK(U+2):U=PEE
K(U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536
20040 RETURN
20099 REM *** SOUND ROUTINES ***
20100 POKEU+1,UN:POKEU+2,UL:US=USR(0):RETURN
20110 FORUZ=1TO4:UN=40:UL=9:GOSUB20100:UN=80:GOSUB20100:NEXT:RET
URN
20120 OUT255,0:OUT255,10:RETURN
20130 UL=3:FOR UN=250TO100STEP-10:GOSUB20100:NEXT:RETURN
20140 FORUZ=1TO30:OUT255,0:OUT255,6:NEXT:RETURN
```

Flash screen routines.

```
29999 REM *** FLASH SCREEN ***
30000 JF="":FORZ=1TO22:READY:JF=JF+CHR$(Y):NEXT
30010 IFPEEK(16396)=201THEN30030 ELSE U=VARPTR(JF):U=PEEK(U+2)*2
56+PEEK(U+1):IFU>32767THENU=U-65536
30020 DEFUSR0=U:RETURN
30030 U=VARPTR(JF):POKE16526,PEEK(U+1):POKE16527,PEEK(U+2):U=PEE
K(U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536
30040 RETURN
```

Get keyboard input.

```
39999 REM *** JOYSTICK SIMULATION ***
40000 I$=INKEY$:IFI$=""THEN40000ELSEI=ASC(I$):RETURN
```

Initialize packed graphics strings.

```
49999 REM *** PACKED STRINGS ***
50000 JR="":FORZ=1TO10:READY:JR=JR+CHR$(Y):NEXT
50010 JD="":FORZ=1TO10:READY:JD=JD+CHR$(Y):NEXT
50020 JB="":FORZ=1TO10:READY:JB=JB+CHR$(Y):NEXT:RETURN
```

TRS-80® SWAT TABLE FOR:

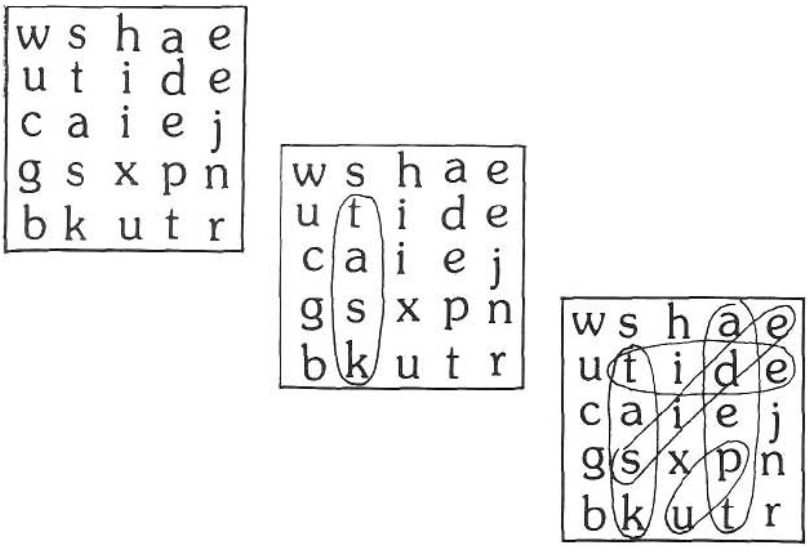
TITAN

LINES	SWAT CODE	LENGTH
1 - 100	IK	515
110 - 230	WR	509
240 - 1020	FN	445
1030 - 1160	QT	402
1170 - 1260	AW	464
1270 - 1360	QY	543
1370 - 2150	CF	491
2160 - 2230	DL	411
2235 - 2400	RE	402
2410 - 2480	AM	457
2500 - 2605	TR	533
2610 - 2999	HH	524
3000 - 3540	EA	449
3550 - 4180	DK	527
4190 - 5060	BZ	464
5070 - 6999	KK	416
7000 - 8560	PC	369
8565 - 8999	RX	468
9000 - 10000	AW	516
10010 - 10060	GV	537
19999 - 20140	QD	434
29999 - 50020	KJ	433

NOTES:

Word Search Puzzle Generator

by David W. Durkee



TRS-80 version by Jon R. Voskuil
***Word-Search Puzzle Generator* is for a TRS-80 with 16K 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 TRS-80 does the rest, placing your words in random orientations in the 32-by-14 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        TRS-80 BASIC         SS
SS    'Word Search Puzzle'     SS
SS  Author:  David W. 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.

```
5 CLEAR 500
10 CLS: PRINT@76, CHR$(23);"WORD SEARCH PUZZLE": PRINT@396, "BY
DAVID W. DURKEE": PRINT@518,"TRANSLATED BY JON VOSKUIL": PRINT@
716, "COPYRIGHT (C) 1981"
20 FOR I=1 TO 1000: NEXT I
40 CLS: PRINT CHR$(23);"TO CREATE A PUZZLE, SIMPLY TYPE IN A WOR
D AFTER THE '?' PROMPT, AND PRESS 'ENTER'.  WHEN YOU'RE FINISHED
, ENTER 'STOP' AS YOUR  LAST WORD, AND THE COMPUTER WILLDO THE R
EST."
50 PRINT:PRINT"PLEASE CHOOSE:": PRINT"  1 - FOR NORMAL DISPLAY D
URING        ENTRY": PRINT"  2 - FOR BLANK DISPLAY (SO THAT
  YOU CAN'T SEE THE PUZZLE)"
55 K$=INKEY$: K=VAL(K$): IF K<1 OR K>2 THEN 55
60 DEFINT A,B: DIM W$(200), A(32,14), B(3,3)
80 CLS:PRINT CHR$(23);
```

Beginning of word-entry loop.

```
90 Z=Z+1
92 PRINT@ 896, "WORD #";Z;: INPUT A$: IF A$="" THEN 92
```

```
95 W$(Z)=A$
100 IF A$="STOP" THEN 500
```

Choose random starting location.

```
120 U=RND(15): L=RND(32)
```

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

```
160 FOR X=-1 TO 1: FOR Y=-1 TO 1
170 IF X=0 AND Y=0 THEN 270
180 X1=L: Y1=U
190 FOR C=1 TO LEN(A$)
200 X1=X1+X: Y1=Y1+Y
210 IF X1>32 OR X1<1 OR Y1>14 OR Y1<1 THEN B(X+2,Y+2)=0: GOTO 27
0
220 IF A(X1,Y1)=0 THEN 250
230 IF A(X1,Y1)<>ASC(MID$(A$,C,1)) THEN B(X+2,Y+2)=0: GOTO 270
240 B(X+2,Y+2)= B(X+2,Y+2)+1
250 NEXT C
260 B(X+2,Y+2)=B(X+2,Y+2)+1: B=B+1
270 NEXT Y,X
280 IF B=0 THEN 120
```

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

```
310 R=2: D=2
320 FOR X=1 TO 3: FOR Y=1 TO 3
330 IF B(X,Y)>B(R,D) THEN R=X:D=Y
340 NEXT Y,X
350 X=R-2: Y=D-2
360 IF X=-1 AND Y=-1 AND B(1,1)=1 THEN 380
370 GOTO 400
380 X=RND(3)-2: Y=RND(3)-2
390 IF (X=0 AND Y=0) OR B(X+2,Y+2)=0 THEN 380
400 X1=L: Y1=U
```

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

```
420 FOR C=1 TO LEN(A$)
430 X1=X1+X: Y1=Y1+Y
440 A(X1,Y1)=ASC(MID$(A$,C,1))
445 IF K=2 THEN 460
450 PRINT@ (Y1-1)*64 + (X1-1)*2, CHR$(A(X1,Y1));
460 NEXT C
```

```
470 B=0: FOR X=1 TO 3: FOR Y=1 TO 3: B(X,Y)=0: NEXT Y,X
480 PRINT@896, STRING$(32,32);: GOTO 90
```

Prepare answer key.

```
500 FOR X=1 TO 32: FOR Y=1 TO 14
510 IF A(X,Y)<>0 THEN 530
520 A(X,Y)=45: PRINT@ (Y-1)*64 + (X-1)*2, "-";
530 NEXT Y,X
540 PRINT@896,"";: INPUT"POSITION PAPER AND HIT ENTER";K$
547 GOSUB 670
550 LPRINT" ":LPRINT"  WORD PUZZLE ANSWER KEY"
560 FOR I=1 TO 31: LPRINT" ":NEXT I
```

Fill in blanks with random letters.

```
570 PRINT@896, "PLEASE WAIT A FEW MOMENTS. . . ";
590 FOR X=1 TO 32: FOR Y=1 TO 14
600 IF A(X,Y)<>45 THEN 620
610 B=RND(26)+64: A(X,Y)=B
620 NEXT Y,X
630 GOSUB 670
640 LPRINT" ": LPRINT"    COMPUTER-GENERATED": LPRINT"    WORD-S
EARCH PUZZLE"
650 FOR I=1 TO 31: LPRINT" ": NEXT I: GOTO 720
```

Print complete puzzle.

```
670 LPRINT" "
680 FOR X=1 TO 32: FOR Y=1 TO 14
690 LPRINT CHR$(A(X,Y));" ";
700 NEXT Y: LPRINT: NEXT X
710 RETURN
```

Print word list.

```
720 LPRINT" ":LPRINT" WORD LIST:": LPRINT" "
730 FOR I=1 TO Z-1: LPRINT W$(I): NEXT I
```

Another copy? If not, then end.

```
760 PRINT@ 896, STRING$(32,32);: PRINT@ 896,"";: INPUT "WOULD YO
U LIKE ANOTHER COPY";K$: IF LEFT$(K$,1)="N" THEN END
770 PRINT@ 896,"";: INPUT"ADVANCE PAPER AND HIT ENTER";K$: GOTO
630
```

TRS-80® SWAT TABLE FOR:
WORD SEARCH PUZZLE GENERATOR

LINES	SWAT CODE	LENGTH
5 - 55	TD	534
60 - 200	BS	267
210 - 340	XM	306
350 - 460	NN	270
470 - 590	SG	371
600 - 720	VY	273
730 - 770	CI	176

NOTES:

compu-sketch

by Roger W. Robitaille Sr.
***Compu-Sketch* is a graphics program for a TRS-80 with 16K RAM.**

Compu-Sketch can help transform anyone into a computer artist. The screen becomes your canvas, and the keyboard your brush. After you complete your masterpiece, you can store it electronically, and preserve it for the world to see.

The program was designed to be easy to use, and provides plenty of prompting to the user through menus. The first menu you encounter allows you to choose among the five major parts of the program: the Draw routine, explained below; the Image Save and Read routines, which work with tape or disk; a printer output routine; and a routine that allows you to review, delete, or modify any of the images in memory.

The draw mode is where most of the interaction with the program takes place, and is where the actual drawing is done. You will see a list of options, at the bottom of the screen, and a flashing cursor, in the middle of the screen. Imagine this cursor as the tip of your paintbrush, which you can move, pick up, and set down to draw.

On the bottom right of the screen are two displays that give you important information. REPT tells you whether the automatic repeat is on or off. When it is on, pressing one of the arrow keys causes the cursor to move continuously until the key is released. When it is off, one keystroke is required for each space you wish the cursor to move. The other message in this area is FUNC, which tells you which function is active: MOVE means that movement of the cursor will not alter the screen display; SET signifies that moving the cursor will leave a line of points; and RESET will cause the cursor to erase any points it falls on. One last function, TYPE, tells the computer to display all keyboard input on the graphics screen instead of interpreting it as a command. Hit the CLEAR key to exit this mode, and return to MOVE.

Another important display is CURSOR. Within each double-wide graphics block, there are three positions that can be separately accessed. The CURSOR display shows which of those three vertically stacked positions is currently being addressed.

The legal commands while in the Draw routine are:

G: Toggles the automatic repeat.

F: Changes the function (SET, RESET, MOVE, or TYPE).

I: Allows you to save the current image.

S: Sets (turns on) the point at the cursor.

R: Resets (turns off) the point at the cursor.

C: Allows you to enter a single character on the screen by typing either the character or the character's ASCII value. Illegal values are ignored.

M: Returns you to the main menu at the start of the program, leaving all stored images intact. Remember to store the image on the screen before doing this, or it will be lost.

Arrow keys: Used to move the cursor around the screen in the desired direction.

Program Notes

SoftSide's original TRS-80 screen-drawing program appeared in its fourth issue (January 1979). It required less than 1K of code, and did a very creditable job for its time. It was structured around SET/RESET graphics. One of its main drawbacks was linked to the nature of TRS-80 graphics themselves. A single pixel (the smallest possible graphics element) is not as wide as it is tall; in fact, it is just about twice as tall as it is wide. Lines a single pixel wide appear twice as thick when drawn horizontally as when drawn vertically. It made for some awkward drawings, to be sure.

The solution was to devise a method to support a cursor two pixels wide and one tall. This approach produces a more symmetric line. This simple concept, however, has some rather complicated consequences.

There are six pixels that occupy the same area of the screen that a single letter would normally occupy. Since *Compu-Sketch* uses them in horizontal pairs, there are only three of the new double pixels in that same area. And, whereas the six normal rectangular pixels can be arranged in 64 different combinations, this program uses only eight of them.

Much of the logic of the program is based on the knowledge that adding or subtracting the proper offset values results in turning on or off the appropriate pixel pair. The top pair can be turned on or off by adding or subtracting 3; similarly, 12 controls the central pair, and 48 the bottom pair.

Variables

A: Used to position VARPTR of A$ in screen storage routine (lines 700-710).
A$: Used to take data from screen into array P$ (see also A).
AX$: Current function (S = Set, R = Reset, M = Move, T = Type).
AY$: Repeat (Y = Yes, N = No).
C$: Code to be sent to the printer to select condensed or normal printing.
CU: Current cursor position.
CU$: Current cursor character.
E: Character present at cursor position.
I, J: Miscellaneous loops.
IM: Image number.
MD: Current mode, based on Y position of current pixel within current cursor position.
N: Maximum number of frames.
P$(i,j): Image storage array.
Q, QQ, Q$, QQ$: Miscellaneous user input.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        TRS-80 BASIC         SS
SS       'Compu-Sketch'        SS
SS By Roger W. Robitaille Sr. SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
100 CLS:CLEAR50:IFMEM$.8)32767THENCLEAR32767ELSECLEAR(MEM$.8)
102 N=FRE(A$)/900:DEFINTD-K:DIMP$(12,N)
105 FORI=1TON:P$(0,I)="UNUSED":NEXTI
```

```
110 AX$="M":AY$="N":CU=415:CU$=CHR$(140):MD=2:E=128

120 CLS:PRINT@20,"MENU":PRINT"DRAW MODE       1":PRINT"DISK SAVE
      2":PRINT"DISK READ       3":PRINT"PRINTER OUTPUT  4":PRIN
T"IMAGE REVIEW    5":PRINT"":INPUT"SELECTION";Q:CLS:ONQGOSUB490,
300,350,5800,900

122 GOTO120

199 SAVE "GRAPHIC/BAS":STOP

200 IFE<129THENE=ASC(CU$):RETURNELSEY=ASC(CU$):IF(Y=E)+(E=191)TH
ENRETURN

201 IF((Y=131)*(E=140))+((Y=140)*(E=131))THENE=143:RETURN

202 IF((Y=131)*(E=176))+((Y=176)*(E=131))THENE=179:RETURN

203 IF((Y=131)*(E=188))+((Y=140)*(E=179))+((Y=176)*(E=143))THENE
=191:RETURN

204 IF((Y=140)*(E=176))+((Y=176)*(E=140))THENE=188:RETURN

205 RETURN

210 IFE<129THENE=128:RETURNELSEY=ASC(CU$):IFY=ETHENE=128:RETURN

211 IF(Y=131)*((E=140)+(E=176)+(E=188))THENRETURN

212 IF(Y=140)*((E=131)+(E=176)+(E=179))THENRETURN

213 IF(Y=176)*((E=131)+(E=140)+(E=143))THENRETURN

214 IFY=131THENE=E-3ELSEIFY=140THENE=E-12ELSEIFY=176THENE=E-48

215 RETURN

300 PRINT@832,;:INPUT "DISK OR TAPE";Q$:Q$=LEFT$(Q$,1):IF Q$="D"
THENPRINT@896,"WHAT IS THE FILE NAME TO BE STORED";:ELSEPRINT@89
6,"HIT ENTER WHEN TAPE IS READY TO SAVE";

310 INPUTQQ$:IF Q$="D" THEN CLOSE:OPEN"O",1,QQ$

320 FORIM=1TON:FORI=0TO12

325 IFP$(0,IM)="UNUSED"THENI=12:NEXTI,IM:IFQ$="D"THENCLOSE:RETUR
NELSEPRINT#-1,CHR$(34);"END OF DATA";CHR$(34):RETURN

330 IF Q$="D" THEN PRINT#1,CHR$(34);P$(I,IM);CHR$(34):NEXTI,IM:C
LOSE:RETURN

332 PRINT#-1,CHR$(34);P$(I,IM);CHR$(34):NEXTI,IM:PRINT#-1,CHR$(3
4);"END OF DATA";CHR$(34):RETURN

350 PRINT@0,CHR$(31);:INPUT "DISK OR TAPE";Q$:Q$=LEFT$(Q$,1):IFQ
$="D"THENPRINT@0,"WHAT IS THE FILE NAME TO BE RECALLED";:ELSEPRI
NT@0,"HIT ENTER WHEN TAPE IS READY TO LOAD";

360 INPUT QQ$:PRINT@64,STRING$(64,"-"):IM=1:I=0:IFQ$="D"THENCLOS
E:OPEN"I",1,QQ$

370 IF Q$="D" THEN IFEOF(1)THENRETURN:ELSEINPUT#1,P$(I,IM):GOTO
374
```

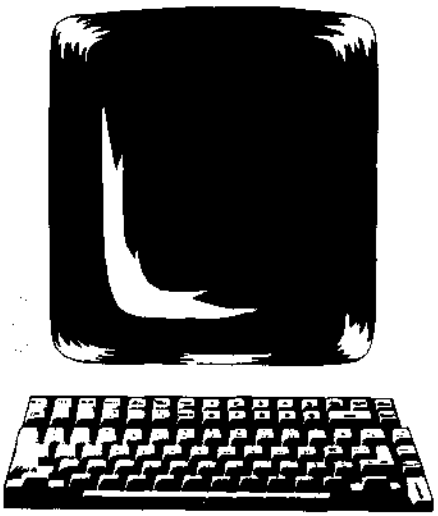
```
372 INPUT#-1,P$(I,IM):IFP$(I,IM)="END OF DATA"THENP$(I,IM)="UNUS
ED":RETURN
374 PRINT@128+I*64,CHR$(31);P$(I,IM);:I=I+1:IFI=13THENI=0:IM=IM+
1
376 GOTO 370
490 PRINT@768,STRING$(64,"-");:GOSUB600
500 PRINT@CU,CU$;:Q$=INKEY$:IF Q$=""ANDAX$()"T"ANDAY$="Y"THENIF(
PEEK(14400)AND8)THENQ$=CHR$(91)ELSEIF(PEEK(14400)AND16)THENQ$=CH
R$(10)ELSEIF(PEEK(14400)AND32)THENQ$=CHR$(8)ELSEIF(PEEK(14400)AN
D64)THENQ$=CHR$(9)
503 POKE 15360+CU,E:IF Q$="" THEN 500
504 IFAX$="T"THENIFQ$=CHR$(31)THENAX$="M":GOSUB610:ELSEIFQ$<>CHR
$(10)ANDQ$<>CHR$(9)ANDQ$<>CHR$(8)ANDQ$<>"["THENPRINT@CU,Q$;:E=AS
C(Q$):Q$=CHR$(9)
505 IFQ$="C"THENMD=MD-1:GOSUB590
507 IFQ$=CHR$(10)THENMD=MD+1:GOSUB590
510 IF(Q$=CHR$(8))*(CU>0)THENCU=CU-1:E=PEEK(15360+CU)
515 IF(Q$=CHR$(9))*(CU<767)THENCU=CU+1:E=PEEK(15360+CU)
520 IF(Q$="S")+((AX$="S")*((Q$="[")+(Q$=CHR$(8))+(Q$=CHR$(9))+(Q
$=CHR$(10))))THENGOSUB200:POKE15360+CU,E
525 IF(Q$="R")+((AX$="R")*((Q$="[")+(Q$=CHR$(8))+(Q$=CHR$(9))+(Q
$=CHR$(10))))THENGOSUB210:POKE15360+CU,E
530 IFQ$="G"THENIFAY$="N"THENAY$="Y":GOSUB600:ELSEAY$="N":GOSUB6
00
535 IFQ$="F"THENPRINT@832,CHR$(31);:PRINT@896,"FUNCTION:--> SET
= S, RESET = R, MOVE = M, TYPE = T";:INPUTAX$:GOSUB600
540 IFQ$="I"THENGOSUB690
545 IFQ$="C"THENGOSUB650
550 IFQ$="M"THENRETURN
580 GOTO500
590 PRINT@CU,CHR$(E);:IF(MD=0)*(CU>63)THENMD=3:CU=CU-64:ELSEIF(M
D=4)*(CU<703)THENMD=1:CU=CU+64:ELSEIF(CU<64)*(MD=0)THENCU=704+CU
:MD=3ELSEIF(MD=4)*(CU>704)THENMD=1:CU=CU-704
591 E=PEEK(15360+CU)
595 IFMD=1THENCU$=CHR$(131)ELSEIFMD=2THENCU$=CHR$(140)ELSEIFMD=3
THENCU$=CHR$(176)
596 PRINT@941,CU$;:RETURN
600 PRINT@832,CHR$(31);:PRINT@832,"G = REPEAT (ON/OFF)":PRINT"F
= FUNCTION SELECT":PRINT"I = SAVE/SHOW IMAGE";:PRINT@852,"S = SE
T CURSOR";:PRINT@916,"R = RESET CURSOR";
602 PRINT@980,"[";CHR$(94);CHR$(92);CHR$(93);"= CONTROLS";:PRINT
@933,"CURSOR ";CHR$(191);CU$;
605 PRINT@869,"C = SET CHAR.";:PRINT@997,"M = MAIN MENU";
610 PRINT@883,CHR$(149);"FUNC >";:IF AX$="T"THENPRINT"TYPE ";:EL
SEIFAX$="S"THENPRINT"SET  ";:ELSEIFAX$="R"THENPRINT"RESET";:ELSE
```

```
PRINT"MOVE ";
615 PRINT@947,CHR$(149);"REPT >";:IF AY$="Y" THEN PRINT"YES";:EL
SEPRINT"NO ";
618 PRINT@1011,STRING$(11,131);
620 RETURN
650 PRINT@832,CHR$(31);:PRINT@896,"CHARACTER";:INPUTQ$:IF LEN(Q$
)>1 THEN IF VAL(Q$)<32 THEN Q$=LEFT$(Q$,1): ELSE Q=VAL(Q$):IF Q<
=128 OR Q=131 OR Q=140 OR Q=143 OR Q=176 OR Q=188 OR Q=191 OR Q=
179 THEN Q$=CHR$(Q):ELSE 650
652 GOSUB600:PRINT@CU,Q$;:E=PEEK(15360+CU):RETURN
680 FORIM=1TON:IFP$(1,IM)=""THENRETURNELSENEXTIM:RETURN
690 GOSUB680:PRINT@832,CHR$(31);:IFIM<=NTHENPRINT@832,"THE NEXT
AVAILABLE IMAGE # IS";IM:INPUT"WHAT IS THE IMAGE TITLE (8 CHRS)"
;Q$:Q$=LEFT$(Q$,8):ELSEINPUT"THERE IS NO UNUSED IMAGE SPACE.  HI
T ENTER";Q:GOSUB600:RETURN
695 FORJ=1TON:IFP$(0,J)=Q$THENPRINT@896,"NAME ALREADY IN USE, <R
> REPLACE, <N> NEW IMAGE NAME";:INPUTQQ$:IFQQ$="N"THENGOTO690ELS
EIM=JELSENEXTJ
697 P$(0,IM)=Q$
700 A$="":FORI=0TO11:PRINT@999,"LINE";I+1;
705 A=VARPTR(A$):POKEA,64:POKEA+1,(I-INT(I/4)*4)*64:POKEA+2,60+I
NT(I/4)
710 P$(I+1,IM)=A$:NEXTI:GOSUB600:RETURN
900 CLS:MK$="##  %      %     ":PRINT@26,"IMAGE INDEX"
902 FOR I=1TON:PRINTUSINGMK$;I,P$(0,I),:NEXTI
904 PRINT@832,;:INPUT"WHICH IMAGE # DO YOU WISH TO VIEW";IM
905 GOTO 2000
906 PRINT@832,CHR$(31);"DO YOU WANT TO VIEW ANOTHER IMAGE (Y/N)"
;:INPUT Q$:IF LEFT$(Q$,1)="Y" THEN 900
910 RETURN
2000 Q$="":PRINT@0,;:FORI=1TO12:PRINTP$(I,IM);:NEXTI:PRINTCHR$(3
1);:PRINT@896,"PRESS <E> TO ERASE, <M> TO MODIFY, <ENTER> TO CON
TINUE";:INPUTQ$:IFQ$=""THEN 906
2005 IF Q$="M" THEN 490
2010 P$(0,IM)="UNUSED":FORI=1TO12:P$(I,IM)="":NEXTI:GOTO 906
5800 PRINT:PRINT"CONDENSED REPORT - 1, NORMAL REPORT - 2, MENU -
 3";:INPUTQ1:IF Q1<>1 AND Q1<>2 THEN RETURN
5802 C$=""
5805 PRINT "ENTER TYPE OF PRINTER YOU HAVE:":PRINT"MICROLINE 80
   1":PRINT"MX-80           2":PRINT"OTHER           3":PRINT:PR
INT"SELECTION";:INPUT Q
5810 IFQ1=1THENIFQ=1THENC$=CHR$(29):ELSEIFQ=2THENC$=CHR$(15):ELS
EIFQ=1THENC$=CHR$(30)ELSEIFQ=2THENC$=CHR$(18)
5850 PRINT:INPUT"WHAT IMAGE WOULD YOU LIKE TO PRINT";X
6000 LPRINTC$:FORI=0TO12:LPRINTP$(I,X):NEXTI:RETURN
```

TRS-80® SWAT TABLE FOR:
COMPU-SKETCH

LINES	SWAT CODE	LENGTH
100 - 203	BA	562
204 - 325	PH	550
330 - 490	XV	515
500 - 525	UN	583
530 - 600	MH	580
602 - 652	OQ	514
680 - 900	KI	507
902 - 5805	GY	598
5810 - 6000	DG	147

NOTES:



RANDOM ACCESS DATABASE

by Mark Pelczarski

TRS-80 random-access version by Robert Jacobs
***Data Base* is a data-management program for a TRS-80 with 48K RAM and a disk drive. A printer is highly desirable.**

Running the *Data Base*

The first choice you will need to make is whether to initialize a new file or load an existing file. Specify this choice by pressing "L" or "I". 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. Any time you want to create a new file with a different type of data, 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 are: 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 be usually stored in this type of database format.

To initialize a new file, give your file a name. Don't specify an extension, as the program requires and supplies its own special extensions. Then tell the computer how many headings you want and their names. An example would be a file named "Address," 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. You might use "Computer" for your seventh heading, so you would know what kind of computer a particular person owns. Your entire record may be no larger than a total of 256 bytes, due to the limitations of the TRS-80's random-access files.

Your data will be organized into a table. The headings will be your column headings, and each row will have one set of information across those headings. A set of such information is called a record. Once a file has been created, any time that you use the database you only have to give the file name ("Address" is our example) and all of the information will automatically be loaded from the disk.

To load a file, press "L" rather than

"I". You will then see a list of all the data files on your diskette. Type the name (omit the extension) of the file you wish to load.

The Main Options

After initialization or loading, you will be given a list of choices for manipulating your database. Here are the choices:

(S) SAVE current data
(P) PRINT data [to screen or printer]
(A) ADD a record
(C) CHANGE a record [such as an address change]
(D) DELETE a record
(T) SORT data
(F) FILE [diskette directory]
(N) NEW data file [equivalent to quitting and re-running program]
(Q) QUIT

Adding A Record

This is your logical first choice, since, with no data in memory, the other options aren't too much fun. 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. Note that commas and semicolons don't work in the data. Also see the note below about searching and sorting numeric fields, if you plan to do such.

Printing A Record

To see if your data is really there, type "P" to print your record. The program will then show you a menu of printing options. After that choice, a list of headings will be displayed, followed by the choices "BEGIN" and "Return to Menu." Choose the number next to the word "BEGIN" and press RETURN. Each record that 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 M 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 of 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 next to 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, 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 want NAME, < ,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, > ,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, = ,60*. The asterisk says that anything may follow. This is also an easy way to find records knowing exact information. If you can't spell "Pelczarski" (or if you don't like typing) 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 can specify several search criteria. You might, for example, want to find everyone in your list whose ZIP code starts with a "60" and

who owns a TRS-80. You would specify ZIP CODE,=,60*, then specify COMPUTER,=,TRS-80, and tell it to begin. The program will ask if the item must meet *all* of the conditions, or *any* of the conditions. "All" would find only those with ZIP 60 who also own a TRS-80. "Any" would find everyone with ZIP 60, plus everyone owning a TRS-80 (technically, everyone with ZIP 60, *or* owning a TRS-80). Up to eight such criteria may be specified, so you may look for everyone whose name starts with D through F (>D and < F), whose ZIP starts with 9, and who owns an Apple and a TRS-80, etc.

Changing Records

To change a record, choose (C) from the options page. After specifying whatever search criteria you want, the appropriate record(s) will be shown on the screen. The items under each heading will then be shown in sequence, and the program will wait for you to type "K" to keep, "C" to change, or "R" if the remainder of the record is okay. If you type "C", you'll be asked for the information with which to replace the old item.

Deleting Records

After choosing (D) from the options page and going through the search steps, you'll be asked to verify that you want each particular record deleted. Type "Y" to delete. Once it's deleted, it's gone.

Saving a File

When you quit, your file is saved automatically. Type (Q) to quit. Typing (S) from the options page saves your current file on disk without quitting the program.

Sorting

The (T) option 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. The latter means that numbers don't always sort the way you want. 125, 34, and 7 will come out in that order because numbers are sorted according to the ASCII code of the first character in each. To get a true numeric sort, add leading zeros to the maximum number of places, such as 007, 034, and 125. This will force proper sequencing.

Files

If your DOS permits access to directories from BASIC, you can get a directory directly from the options page by typing (F) if you have deleted the apostrophes from lines 270, 380, and 600. Check your DOS manual for the proper form of the directory command. Be sure to check your typing with *SWAT* before you make any changes.

Switching Data Files

You can load or create a new file without rerunning the program by selecting (N) from the options. Be sure to verify that the current file has been saved.

Formatting Output

The formatter in this *Data Base* may be one of the most versatile ones around. Although there are still a few things it can't do, it does a lot that many "professional" databases don't allow. You can specify the exact form in which you want each record printed. Each record is printed in sequence, meaning that you cannot mix records across a page. You can specify which headings are to be printed and where, which items are to be printed and where, and what (if any) additional character strings should be printed on the form. You may want to include your company name, an expanded version of a heading instead of the heading itself, or just some lines to separate items.

To create a format, choose the special format option when printing. You'll be asked if you want to load or create one. 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 "Address" 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 are used in this format, and neither is the phone number.

Another example is a format that will print a separate little form for each person in the database. For lack of a better example, I'll have the following printed:

THE FOLLOWING PERSON OWNS A

TRS-80

NAME JOE TATE
PHONE 555-1212

Here's the format to do it:

String,----------
Next line, 1
String, THE FOLLOWING PERSON OWNS AN
Next Line, 2
Tab, 9
Item, Computer
Next Line, 2
Heading, Name (Type "1" for hdg. #)
Tab, 7
Item, Name (Type"1" for item #)
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 last, so I can just end the format after printing the last item.

That's all there is to formatting. Play around with it a little to see what it does for you. After a format is created, you'll be asked to name it, and it will automatically be saved to disk. In the future, you'll be able to load it back in when you need it. The names for formats work the same way as the names for data files, except that the format files all have the extension "/FMT".

A Few Final Words

There are a lot of things this *Data Base* program still cannot do, but it is a good introduction for those of you

who don't know all of what a database program can be used for. As yet, it has no real numeric capability; it doesn't take advantage of disk capabilities; and some of the routines are slow. On the plus side, it has a lot of the features you should look for in a database, and if you ever decide to shop for one (they're expensive) you'll have an idea of the features to consider. I'm amazed that there are $200 database programs out there that don't even have basic sorting functions; and most only have a rather primitive print formatting.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        TRS-80 BASIC         SS
SS        'Data  Base'         SS
SS  Authors: Mark Pelczarski   SS
SS      and Robert Jacobs      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 control routine

```
100 CLS:CLEAR20000
102 SC=16414:S1=PEEK(SC):S2=PEEK(SC+1)
103 PR=16422:P1=PEEK(PR):P2=PEEK(PR+1)
105 DIMC$(7),C1%(7),C2%(7),F$(5):CH=0
107 DEFFNSB$(A$)=LEFT$(A$+"  ",INSTR(A$+"  ","  ")-1)
110 PRINT"(I) INITIALIZE A NEW DATA SET"
120 PRINT"(L) LOAD A PREVIOUSLY SAVED DATA SET";
130 GOSUB60000
140 IFA$="L"ORA$=CHR$(108)GOSUB1000:GOTO200
150 IFA$="I"ORA$=CHR$(105)GOSUB1500:GOTO200
160 GOTO100
200 CLS:PRINT"(S) SAVE CURRENT DATA"
210 PRINT"(P) PRINT DATA"
220 PRINT"(A) ADD DATA"
230 PRINT"(C) CHANGE A RECORD"
240 PRINT"(D) DELETE A RECORD"
260 PRINT"(T) SORT"
270 'PRINT"(F) FILE NAMES":REM OMIT APOSTROPHE IF YOUR DOS HAS A
 METHOD TO GET A DIRECTORY FROM BASIC
280 PRINT"(N) NEW DATA FILE"
290 PRINT"(Q) QUIT"
```

```
297 PRINT:PRINT NI;" RECORDS FULL, ROOM FOR ";MX-NI-1;" MORE"
300 GOSUB 60000:PRINT
310 IFA$="S"ORA$=CHR$(115)GOSUB2000:GOTO200
320 IFA$="P"ORA$=CHR$(112)GOSUB3000:GOTO200
330 IFA$="A"ORA$=CHR$(97)GOSUB4000:GOTO200
350 IFA$="C"ORA$=CHR$(99)THENSB=3:GOSUB8000:GOTO200
360 IFA$="D"ORA$=CHR$(100)THENSB=4:FS=1:GOSUB8000:GOTO200
370 IF A$="T"ORA$=CHR$(116)THEN GOSUB7000:GOTO200
380 '1F A$="F"ORA$="F"THENGOSUB600:GOTO200:REM OMIT APOSTROPHE I
F YOUR DOS HAS A METHOD TO GET A DIRECTORY FROM BASIC
390 IFA$="Q"ORA$=CHR$(113)ORA$="N"ORA$=CHR$(110)THEN500
410 GOTO200
500 IFSS=1THEN530
520 GOSUB2000
530 CLOSE2
540 IFA$="N"ORA$=CHR$(110)THEN100
550 END
600 'CMD"DIR":GOSUB60000:RETURN:REM REMOVE APOSTROPHE AND CHANGE
 CMD"DIR" TO THE APPROPRIATE DIRECTORY COMMAND FOR YOUR DOS IF I
T HAS A METHOD TO DO THIS FROM BASIC
```

Load subroutine.

```
1000 INPUT"FILE NAME (NO EXTENSION PLEASE) ";F$
1010 ONERRORGOTO1230
1020 OPEN"I",1,F$+"/HDG"
1030 INPUT#1,NH,NI,MX,LK,FF
1040 ONERRORGOTO0
1130 DIMH$(NH),B%(NH+1),I$(MX),P%(MX),TI$(NH),B$(NH+1,FF)
1140 FORI=0TONH:INPUT#1,H$(I),B%(I):NEXT
1150 INPUT#1,B%(NH+1)
1160 IFNI=0THEN1180
1170 FORI=1TONI:INPUT#1,P%(I):NEXT
1180 CLOSE1
1190 GOSUB20000
1200 IFNI=0THEN1220
1210 GOSUB1300
1220 SS=1:RETURN
1230 PRINT"FILE NOT FOUND";:GOSUB60000:RESUME100
1300 PRINT:FORI=1TONI
1305 PR%=INT((P%(I)-1)/FF)+1
1310 GET 2,PR%
1320 I$(I)=FNSB$(B$(CH,P%(I)-FF%(PR%-1)))
1330 NEXT
1340 RETURN
```

Initialize subroutine.

```
1500 INPUT"FILE NAME (NO EXTENTION PLEASE) ";F$
1510 IFF$=""THEN1500
1520 INPUT"HOW MANY HEADINGS";NH
1530 IFNH<1THEN1520
1540 NH=NH-1:NI=0:LK=0:T%=0
1560 DIMH$(NH),B%(NH+1),TI$(NH):B%(0)=0
1570 FORI=0TONH
1580 PRINT"HEADING #";I+1;:INPUT " : ";H$(I)
1590 INPUT"MAXIMUM LENGTH : ";B%(I):T%=T%+B%(I)
1610 NEXT
1615 FF=INT(256/T%):DIMB$(NH+1,FF)
1620 INPUT"WHICH HEADING IS THE LONGEST ON WHICH YOU WILL SORT";
J:J=J-1:IFJ<00RJ>NHTHEN1620
1630 MX=INT((FRE(A$)-2000)/B%(J)+2)
1640 DIMI$(MX),P%(MX)
1650 GOSUB20000
1660 SS=0:RETURN
1700 PRINT:R=P%(I):GOSUB20100
1710 GET 2,PR%
1720 FORJ1=0TONH
1730 TI$(J1)=FNSB$(B$(J1,P%(I)-FF%(PR%-1)))
1740 NEXT
1750 RETURN
1800 PRINT:GOSUB20100
1805 FORJ1=0TONH
1810 LSET B$(J1,R-FF%(PR%-1))=TI$(J1)
1820 NEXT
1830 PUT 2,PR%
1840 RETURN
```

Write subroutine.

```
2000 PRINT:ONERRORGOTO2290
2010 OPEN"D",1,F$+"/HDG"
2020 PRINT#1,NH;NI;MX;LK;FF
2040 FORI=0TONH:PRINT#1,H$(I);",";B%(I):NEXT
2050 PRINT#1,B%(NH+1)
2060 IFNI=0THEN2270
2070 FORI=1TONI:PRINT#1,P%(I):NEXT
2270 CLOSE1
2280 ON ERROR GOTO0:SS=1:RETURN
2290 PRINT"DISK ERROR":GOSUB60000:RESUME200
```

Print subroutine.

```
3000 IFN1=0THENGOSUB9000:RETURN
3005 CLS:PRINT"(A) SCREEN PRINT, DEFAULT FORMAT":PRINT"(B) SCREE
N PRINT, SELECT FORMAT":PRINT"(C) LINE PRINT, DEFAULT FORMAT":PR
INT"(D) LINE PRINT, SELECT FORMAT":IFPEEK(14312)<>63ANDPEEK(293)
<>73THENPRINT:PRINT"PRINTER NOT READY!":GOSUB60000ELSEGOSUB6000
0
3006 IFA$="A"ORA$=CHR$(97)THENFS=1:GOTO3040
3007 IFA$="B"ORA$=CHR$(98)THENGOSUB10000:FS=2:GOTO3040
3008 IFA$="C"ORA$=CHR$(99)THENFS=1:SB=2:LPRINTCHR$(32):LPRINTTIM
E$:GOTO3060
3009 IFA$="D"ORA$=CHR$(100)THENGOSUB10000:FS=2:SB=2:LPRINTCHR$(3
2):LPRINTTIME$:GOTO3060
3010 GOTO3005
3040 SB=1:CLS:PRINT"AFTER EACH RECORD <M> WILL RETURN YOU TO MEN
U":PRINT
3050 PRINT"PRESS ANY KEY . . .":GOSUB60000
3060 GOSUB8010
3090 IFSB=2THENPOKESC,51:POKESC+1,52
3100 RETURN
```

Print one record to screen.

```
3300 ON FSGOSUB 3700,3800
3340 IFSB=1THENGOSUB60000:IFA$="M"ORA$=CHR$(109)THENRS=1
3350 RETURN
```

Print one record default format.

```
3700 CLS:PRINT" ":PRINT"RECORD ";I:PRINT" "
3710 FORJ=0TONH
3720 PRINTH$(J),TI$(J)
3730 NEXTJ
3740 RETURN
```

Print one record custom format.

```
3800 CLS:J=1:T=0:B$=""
3820 J1=VAL(MID$(F$(T),J,1)):J=J+1
3830 IFJ1<5THENN=VAL(MID$(F$(T),J,2)):J=J+2
3840 ON J1 GOTO3850,3860,3870,3890,3910,3970
3850 A$=H$(N):GOTO3950
3860 A$=TI$(N):GOTO3950
3870 B$=LEFT$(B$,N-1):IFLEN(B$)<N-1THENFORJ2=LEN(B$)TON-2:B$=B$+
```

```
" ":NEXT
3880 GOTO3960
3890 PRINTB$:IFN>1THENFORJ2=2TON:PRINT" ":NEXT
3900 B$="":GOTO3960
3910 IFJ>LEN(F$(T))THENT=T+1:J=1
3920 J2=J
3930 IFMID$(F$(T),J2,1)<>"!"THENJ2=J2+1:GOTO3930
3940 A$=MID$(F$(T),J,J2-J):J=J2+1
3950 B$=B$+A$
3960 IFJ>LEN(F$(T))THENT=T+1:J=1
3965 GOTO3820
3970 PRINTB$:RETURN
```

Add subroutine.

```
4000 SS=0:NI=NI+1
4005 CLS:PRINT"RECORD";NI:PRINT
4010 FORJ=0TONH
4020 GOSUB4500
4030 NEXTJ
4040 IFLK=0THENR=NI:GOTO4080
4050 R=LK
4060 GOSUB 20100
4070 GET 2,PR%:LK=CVI(B$(0,R-FF$(PR%-1)))
4080 GOSUB1800:P%(NI)=R:I$(NI)=TI$(CH)
4090 RETURN
4500 PRINTH$(J);" (";B%(J);"CHARACTERS";")";:INPUT" : ";TI$(J)
4510 RETURN
```

Change subroutine.

```
5000 CLS:PRINT"(C) CHANGE ITEM, (K) KEEP ITEM, OR (R) KEEP REMAI
NDER OF RECORD"
5030 PRINT:PRINT"RECORD ";I
5040 CS=1:RS=0:FORJ=0TONH
5050 PRINT:PRINT H$(J);" : ";TI$(J);" ";
5055 IFRS=1THENPRINT:GOTO5090
5060 GOSUB 60000:IFA$<>"C"ANDA$<>CHR$(99)ANDA$<>"K"ANDA$<>CHR$(1
07)ANDA$<>"R"ANDA$<>CHR$(114)THEN5060
5065 PRINTCHR$(ASC(A$)+32*(A$>"Z"))
5070 IFA$="K"ORA$=CHR$(107)THEN5090
5075 IFA$="R"ORA$=CHR$(114)THENRS=1:GOTO5070
5080 GOSUB4500
5085 CS=0
5090 NEXTJ
```

```
5095 RS=0
5100 IFCS=0:R=P%(I):GOSUB1800:I$(I)=TI$(CH)
5110 RETURN
```

Delete subroutine.

```
6000 PRINT:PRINT"DELETE THIS RECORD? ";
6070 GOSUB60000:IFA$<>"Y"ANDA$<>CHR$(121)ANDA$<>"N"ANDA$<>CHR$(1
10)THEN6070
6080 PRINTCHR$(ASC(A$)+32*(A$>"Z")):IFA$="N"ORA$=CHR$(110)THEN61
50
6090 LSET B$(0,P%(I)-FF*(PR%-1))=MKI$(LK)
6095 PUT 2,PR%
6110 LK=P%(I)
6120 FORI1=I+1TONI
6130 I$(I1-1)=I$(I1):P%(I1-1)=P%(I1)
6135 NEXTI1
6140 NI=NI-1:SS=0:I=I-1
6150 RETURN
```

Sort subroutine.

```
7000 IFNI=0GOSUB9000:RETURN
7010 CLS:FORJ=0TONH
7020 PRINT@128+(J*32),"(";J+1;") ";H$(J)
7030 NEXTJ
7040 INPUT "SORT ON WHICH HEADING";J1
7045 J1=J1-1
7050 IFJ1<0ORJ1>NHRETURN
7055 IFJ1<>CHTHENCH=J1:GOSUB1300
7060 PRINT"(A) ASCENDING OR (D) DESCENDING":GOSUB60000
7070 IFA$="A"ORA$=CHR$(97)THENA=1:GOTO7100
7080 IFA$="D"ORA$=CHR$(100)THENA=2:GOTO7100
7090 GOTO7060
7100 IFA=1THENFORI=0TONI-1ELSEFORI=1TONI
7110 T=I
7120 FORI1=T+1TONI
7122 PRINT@960,I;I1;
7125 ONAGOTO7130,7140
7130 IFI$(I1)<I$(T)THENT=I1
7135 GOTO7145
7140 IFI$(I1)>I$(T)THENT=I1
7145 NEXTI1
7150 IFT=ITHEN7180
7155 FORJ=0TONH
```

```
7160 T$=I$(T):I$(T)=I$(I):I$(I)=T$
7170 J1=P%(T):P%(T)=P%(I):P%(I)=J1
7180 NEXTI
7200 SS=0:RETURN
```

Search subroutine.

```
8000 IFNI=0THENGOSUB9000:RETURN
8010 I1=1:I2=NI:J=0:C1%(0)=-1:BS=1
8015 CLS:PRINT"SEARCH CRITERIA:":PRINT
8020 PRINT" 0 ) RECORD NUMBER"
8030 FORI=0TONH:PRINT@128+((I+1)*32),I+1;") "H$(I):NEXTI
8035 PRINT:PRINTNH+2;") BEGIN"
8036 PRINTNH+3;") RETURN TO MENU"
8040 PRINT@896,;:INPUT"WHICH FIELD";I:IFI<0ORI>NH+3THEN8040
8045 IFI=NH+2THENC1%(J)=-1:GOTO8150
8046 IFI=NH+3 THEN RETURN
8050 C1%(J)=I-1
8055 CLS:IFI=0PRINT@128,"SELECTING ON RECORD NUMBER":GOTO8060ELS
EPRINT@128,"FIELD SELECTED: ";H$(I-1)
8060 PRINT@256,"INDICATE (1) SMALLER,  (2) EQUAL, OR (3) LARGER"
:GOSUB60000:C2%(J)=VAL(A$):IFC2%(J)<1ORC2%(J)>3THEN8060
8080 PRINT@384,"COMPARED TO: ";:IFC1%(J)=-1THEN8100
8090 INPUT"  ";C$(J):J=J+1:IFJ>7THEN8160
8095 GOTO8015
8100 INPUT"  ";I:IFI<1ORI>NITHEN8100
8110 IFC2%(J)=1THENI2=I
8120 IFC2%(J)=2THENI1=I:I2=I
8130 IFC2%(J)=3THENI1=I
8140 GOTO8015
8150 IFJ<2THEN8200
8160 PRINT@896,"1) ITEM MUST MEET ALL CONDITIONS":PRINT"2) ITEM
MAY MEET ANY CONDITION";:GOSUB60000:BS=VAL(A$):IFBS<1ORBS>2THENB
160
8200 PRINT"SEARCHING . . .":RS=0:J1=C1%(0):IFSB=2THENPOKESC,P1:P
OKESC+1,P2
8210 DS=0:FORJ=0TO7
8220 IFC1%(J)=-1THENJ=7:GOTO8240
8230 IFJ1<>C1%(J)THENJ1=-2
8240 NEXT
8245 IFJ1>-1 ANDJ1<>CHTHENCH=J1:GOSUB1300
8246 IFJ1=-2THENDS=1
8250 I=I1-1:FORI3=I1TOI2:I=I+1
8251 IFDS=0THENTI$(CH)=I$(I):GOTO8255
8252 GOSUB1700
```

```
8255 AS=0:FORJ=0TO7
8260 IFC1%(J)=-1THENJ=7:GOTO8345
8270 ONC2%(J)GOTO8280,8290,8310
8280 IFTI$(C1%(J))<=C$(J)THEN8330
8285 GOTO8340
8290 IFTI$(C1%(J))=C$(J)THEN8330
8295 IFRIGHT$(C$(J),1)<>"*"THEN8340
8298 T=LEN(C$(J))-1:IF LEN(TI$(C1%(J)))<TTHEN8340
8302 IFLEFT$(TI$(C1%(J)),T)=LEFT$(C$(J),T)THEN8330
8305 GOTO8340
8310 IFTI$(C1%(J))>=C$(J)THEN8330
8320 GOTO8340
8330 IFBS=2THENAS=1:J=7
8335 GOTO8345
8340 IFBS=1THENAS=2:J=7
8345 NEXTJ
8350 IFAS=0ANDBS=1THEN8355
8352 IFAS<>1THEN8380
8355 IFDS=0THENGOSUB1700
8360 IFSB<>3THENGOSUB3300
8365 IFSB=3THENGOSUB5000
8370 IFSB=4THENGOSUB6000
8375 IF RS=1 THEN I3=I2
8380 NEXTI3
8390 PRINT:PRINT"THAT'S ALL":FORII=1TO500:NEXT
8400 RETURN
```

Error routine #1.

```
9000 PRINT"THERE'S NO DATA IN MEMORY."
9010 FORI=1TO1000:NEXT:RETURN
```

Print formatting.

```
10000 IFF$(0)=""THEN10040
10010 PRINT"SAME FORMAT?";:GOSUB60000
10020 IFA$="Y"ORA$=CHR$(121)THENRETURN
10030 IFA$<>"N"ANDA$<>CHR$(110)THEN10010
10040 PRINT"(L) LOAD FORMAT, OR (C) CREATE FORMAT";:GOSUB60000
10050 IFA$="C"ORA$=CHR$(99)THEN10200
10060 IFA$<>"L"ANDA$<>CHR$(108)THEN10040
10090 ON ERROR GOTO10170
10100 INPUT"FORMAT NAME:";A$
10110 OPEN"I",3,A$+"/FMT"
10130 INPUT#3,NF
```

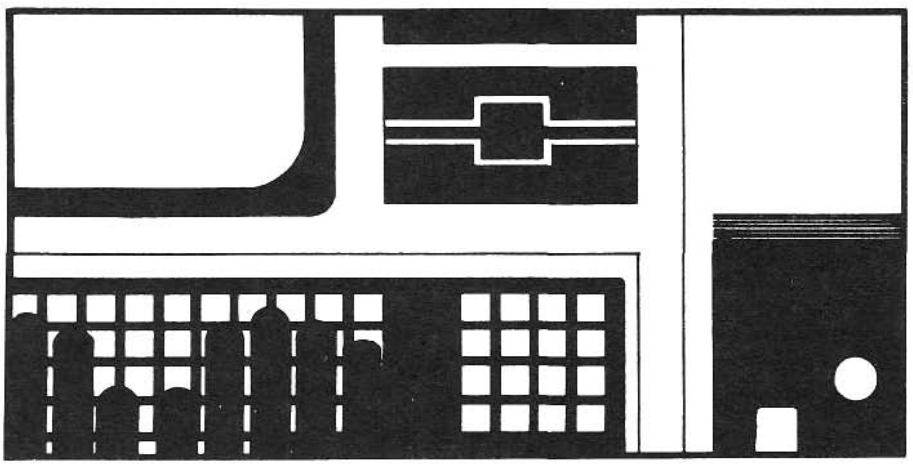
```
10140 FORJ=0TONF:INPUT#3,F$(J):NEXT
10150 CLOSE3
10160 ON ERRORGOTO0:RETURN
10170 PRINT"FORMAT NOT FOUND":GOSUB60000:RESUME200
10200 NF=0:J=0:F$(0)=""
10210 CLS:PRINT"START IN THE UPPER LEFT CORNER AND WORK ACROSS E
ACH LINE"
10220 PRINT"1:HEADING, 2:ITEM, 3:TAB, 4:NEXT LINE, 5:STRING, 6:E
ND":INPUTJ1
10230 IFJ1<1ORJ1>6THEN10220
10240 F$(NF)=F$(NF)+RIGHT$(STR$(J1),LEN(STR$(J1))-1):J=J+1
10250 ONJ1GOTO10260,10260,10300,10300,10350,10400
10260 FORT=0TONH:PRINT@128+(T*32),T+1;") ";H$(T):NEXT
10270 INPUT"WHICH";T:T=T-1:IFT<0ORT>NHTHEN10270
10280 GOTO10310
10300 INPUT"HOW MANY";T:IFT<1ORT>99THENPRINT"OUT OF RANGE.":GOTO
10300
10310 A$=RIGHT$(STR$(T),LEN(STR$(T))-1):IFT<10THENA$="0"+A$
10320 F$(NF)=F$(NF)+A$:J=J+2
10330 GOTO10380
10350 INPUT"STRING:";A$:A$=A$+"!"
10360 IFLEN(A$)+J>255THENNF=NF+1:J=0:F$(NF)=""
10370 F$(NF)=F$(NF)+A$:J=J+LEN(A$)
10380 IFJ>252THENNF=NF+1:J=0:F$(NF)=""
10390 GOTO10210
10400 INPUT"FORMAT NAME:";A$
10405 ON ERROR GOTO10460
10410 OPEN"O",3,A$+"/FMT"
10430 PRINT#3,NF:FORJ=0TONF:PRINT#3,F$(J):NEXT
10440 CLOSE3
10450 ON ERROR GOTO0:RETURN
10460 PRINT"DISK ERROR":GOSUB60000:RESUME10400
20000 ON ERROR GOTO 20070
20005 OPEN"R",2,F$+"/DAT"
20010 FORJ=1TOFF
20020 D%=0
20030 FORI=0TONH
20040 FIELD 2, D%+(J-1)*T% AS DU$,B%(I) AS B$(I,J)
20050 D%=D%+B%(I)
20060 NEXT I:T%=D%:NEXTJ
20065 ON ERROR GOTO 0:RETURN
20070 PRINT"DISK OR FIELD ERROR; PRESS A KEY TO TRY AGAIN":GOSUB
60000:RESUME 100
20100 PR%=INT((R-1)/FF)+1:RETURN
60000 A$="":A$=INKEY$:IFA$=""THEN60000ELSEPRINT:RETURN
```

TRS-80® SWAT TABLE FOR:
DATABASE

LINES	SWAT CODE	LENGTH
100 - 200	MD	355
210 - 320	OD	389
330 - 550	VM	381
600 - 1180	OK	423
1190 - 1500	TR	229
1510 - 1630	DF	363
1640 - 1810	EO	201
1820 - 2280	VG	197
2290 - 3040	LI	501
3050 - 3740	EO	214
3800 - 3920	NE	305
3930 - 4040	EI	215
4050 - 5055	SZ	323
5060 - 6070	NS	325
6080 - 7020	QQ	264
7030 - 7120	KD	263
7122 - 7200	KV	209
8000 - 8055	NG	393
8060 - 8200	BY	474
8210 - 8270	IE	251
8280 - 8340	VI	269
8345 - 9000	PL	207
9010 - 10130	SZ	300
10140 - 10270	FI	421
10280 - 10405	OY	322
10410 - 20050	DB	235
20060 - 60000	AE	154

NOTES:

MICROTEXT 1.2



by Jon Voskuil

"*Microtext 1.2*" is a word processor program for a 16K TRS-80 (Model I or III).

Upon running, *Microtext 1.2* displays a mostly blank screen with an instruction summary line at the top or bottom. You can either start typing, or load a previously saved file from disk or tape. Capitalization of characters on the Model I is accomplished by preceding individual characters to be capitalized by an "@" symbol, and words to be capitalized by a double "@@". Normal use of the Model I's shift key inserts the "@" in front of individual characters; the "@" key itself must be used to insert the double symbol necessary for word capitalization. Pressing and releasing clear, 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 CLEAR-Q will quit the program.

Saving and loading files is simply a matter of answering the questions

```
210 PRINT@ 64, STRING$(63,"-");
220 PRINT @P,"";
```

Input loop.

```
499 ' INPUT LOOP
500 PRINT CU$;
505 C$=INKEY$: IF C$="" THEN 505 ELSE C=ASC(C$)
520 IF C=RTN THEN C$=CR$
530 IF C>90 THEN PRINT B$;"@ ";: L$(LN)=L$(LN)+"@": CHAR=CHAR+1:
 C$=CHR$(C-32): GOTO 740
639 ' BACKSPACE
640 IF C=BKSP THEN IF CHAR<2 THEN 505 ELSE PRINT B$;B$;: CHAR=CH
AR-1: L$(LN)=LEFT$(L$(LN), LEN(L$(LN))-1): GOTO 500
719 ' CTRL CHARACTER
720 IF C=CLR THEN 2000
739 ' END OF LINE
740 CHAR=CHAR+1: IF CHAR>=LWID AND C<>SPC AND C<>RTN THEN GOSUB
1000
759 ' ADD TO STRING
760 L$(LN)=L$(LN)+C$
779 ' RETURN
780 IF C=RTN THEN PRINT B$;C$: LN=LN+1: L$(LN)="": SL=0: CHAR=1:
 GOTO 500
879 ' PRINT ON SCREEN
880 PRINT B$;C$;
899 ' UPDATE SPC POINTER, CHK END OF LINE
900 IF C<>32 THEN 500 ELSE SL=CHAR: IF CHAR=LWID THEN LN=LN+1: L
$(LN)="": CHAR=1: SL=0: PRINT
920 GOTO 500
```

Subroutine to break line at a space and to initiallze the next line.

```
999 ' JUSTIFY AND INCR LINE
1000 IF SL=0 THEN PRINT: GOTO 1100
1020 SS=LWID-SL: FOR J=1 TO SS: PRINT B$;: NEXT
1040 PRINT STRING$(SS,32);:PRINT
1050 IF SL=LWID-1 THEN 1100
1060 L$(LN+1)= RIGHT$(L$(LN), LWID-1-SL)
1080 L$(LN)= LEFT$(L$(LN), SL-1)
1100 LN=LN+1
1120 PRINT L$(LN);CU$;
1140 CHAR=LEN(L$(LN))+2
1150 SL=0
1160 RETURN
```

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        TRS-80 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
```

Title page.

```
10 CLS: PRINT CHR$(23)
20 PRINT @198, "M I C R O T E X T   1 . 2"
30 PRINT @462, "BY JON R. VOSKUIL"
40 PRINT @640,"(C) 1982  SOFTSIDE PUBLICATIONS"
50 FOR Z=1 TO 1000: NEXT Z: GOTO 100
```

Mixed-case conversion routine.

```
59 ' LOWERCASE PRINT RTN
60 X$="": IF PP$="" THEN 74
62 LOK=0: LC=-1: FOR K=1 TO LEN(PP$): CC=ASC(MID$(PP$,K,1)): IF
CC=32 THEN LOK=0
64 IF CC<>64 THEN 70
66 IF LC=0 THEN LOK=-1
68 LC=0: GOTO 72
70 LPRINT CHR$(CC-32* (LC AND CC>64 AND CC<91) * -(NOT LOK));:LC
=-1
72 NEXT
74 RETURN
```

Initialization.

```
99 ' INITIALIZATION
100 CLS
110 CLEAR 22000
115 DEFINT A-Z
120 DIM L$(500)
125 ON ERROR GOTO 20000
130 BKSP=8: RTN=13: SPC=32: CLR=31: B$=CHR$(8): CR$=CHR$(140)
140 CHAR=1
145 CU$=CHR$(95)
150 LN=1
180 LWID=62
190 P=128
200 PRINT @0,"SAVE:CLR-S   LOAD:CLR-L   REVW:CLR-R   EDIT:CLR-E
  PRINT:CLR-P"
```

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 ENTER 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 ENTER 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 CLEAR 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 ENTER 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 CLEAR-F (not ENTER unless you want a carriage return in the text itself). The computer will check to see if 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, line spacing, and (in the case of the Model I) the case conversion option. Pressing ENTER in response to the offered options will select the default value indicated. The case conversion option allows the text to be printed from the Model I just as it appears on the screen, in all upper case and with the embedded @ symbols; or else to convert it (a bit sluggishly, using a BASIC routine) to normal mixed case for a final printout.

Program Notes

The DIMension statement in line 120, and the CLEAR in line 110 of the TRS-80 version, reserve memory for the strings which hold the text. The numbers listed work for 48K 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 effecting 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 possible execution speed.

If you have a TRS-80 Model III, some of the listed lines should be changed as follows:

Line 50: Omit ":GOTO 100"
Lines 59-74: Delete
Line 7050: Delete
Line 7130: Change to "LPRINT TAB(LM);P$;"
Line 7140: Delete
Line 7590: Change to "LPRINT TAB(LM);PP$;"
Line 7600: Delete

Be sure to check your typing with *SWAT* before making any modifications to the program.

Subroutine to process command codes.

```
1999 ' PROCESS CTRL CHARACTERS
2000 P=PEEK(16416) + PEEK(16417)*256 - 15360: 'CURSOR PSN
2020 C$=INKEY$: IF C$="" THEN 2020
2030 C=ASC(C$)
2050 IF C=70 THEN RETURN: 'BACK TO EDIT
2100 IF C=82 THEN GOSUB 3000: GOTO 200: 'REVW
2200 IF C=83 THEN GOSUB 4000: GOTO 200: 'SAVE
2300 IF C=76 THEN GOSUB 5000: GOTO 200: 'LOAD
2400 IF C=81 THEN END: 'QUIT
2500 IF C=80 THEN GOSUB 7000: GOTO 200: 'PRINT
2600 IF C=69 AND LN>1 THEN I=LN-1: PP=P: GOSUB 9000: GOSUB 3000:
 GOTO 200: 'EDIT
2800 P=P-1: GOTO 200
```

Subroutine to review text.

```
2999 ' REVIEW TEXT
3000 CLS: PRINT@ 21, "PRESS ANY KEY TO PAUSE": PRINT STRING$(63,
"-")
3040 IF LN=1 THEN 3210
3050 FOR I=1 TO LN-1
3055 FOR Z=1 TO 20: NEXT Z
3060 PRINT L$(I)
3070 IF INKEY$="" AND NOT STP THEN 3200
3080 STP=0
3090 PP=PEEK(16416) + PEEK(16417)*256 - 15360
3110 PRINT@ 0, "   ENTER: CONTINUE       SPACEBAR: STEP 1 LINE
    E:EDIT     "; STRING$(63,"-");
3120 X$=INKEY$: IF X$="" THEN 3120
3130 X=ASC(X$): IF X=69 THEN GOSUB 9000: GOTO 3000
3160 IF X=13 THEN 3190
3170 IF X<>32 THEN 3110
3180 STP=-1
3190 PRINT @PP,"";
3200 NEXT I
3210 PRINT L$(LN);
3220 P=PEEK(16416) + PEEK(16417)*256 - 15360
3230 RETURN
```

Subroutine to save to tape or disk.

```
3999 ' SAVE TO DISK/TAPE
4000 PRINT@64,STRING$(63,"-");: PRINT@ 0,STRING$(63,32);: PRINT@
 0, "SAVE TO TAPE OR DISK? (T/D/CLEAR) ";
```

```
7530 LP=LEN(P$): IF LP<2 THEN PP$="": P$="": GOTO 7590
7540 PP$=LEFT$(P$,LP-1): P$="": GOTO 7590
7550 C$=MID$(P$,L,1): IF C$=" " THEN 7580
7560 L=L-1: IF L>0 THEN 7550
7570 L=LL
7580 PP$=LEFT$(P$,L): P$=RIGHT$(P$,LEN(P$)-L)
7590 LPRINT TAB(LN);: IF UC THEN LPRINT PP$;: GOTO 7610
7600 GOSUB 60
7610 FOR J=1 TO LS: LIN=LIN+1: LPRINT"": NEXT J
7615 IF LIN>59 THEN FOR J=1 TO 66-LIN: LPRINT"": NEXT J: LIN=0
7620 IF LEN(P$)>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 ' TEXT REJUSTIFY RTN
8000 CLS: PRINT"RE-JUSTIFYING TEXT. . ."
8010 LIN=EL: LN=LN-1
8020 P$="": CR=0: I=EL-1
8030 I=I+1: P$=P$+L$(I)
8040 IF RIGHT$(P$,1)=CR$ THEN CR=-1: GOTO 8060
8050 IF LEN(P$)+LEN(L$(I+1))<256 AND I<LN THEN 8030
8060 GOSUB 8500
8070 IF NOT CR THEN 8100
8079 ' END OF PARAGRAPH
8080 X=I+1-LIN: IF X=0 THEN 8150
8089 ' MOVE LNS DOWN IN ARRAY
8090 FOR J=I+1 TO LN: L$(J-X)=L$(J): NEXT J: LN=LN-X: GOTO 8150
8100 IF I<LN THEN 8030
8110 L$(LIN)=P$
8120 CHAR=LEN(P$)+1: SL=LEN(P$)
8130 FOR I=LIN+1 TO LN: L$(I)="": NEXT I
8140 LN=LIN
8150 RETURN
8500 L=LWID
8510 IF LEN(P$)>LWID THEN 8550
8520 IF NOT CR THEN 8630
8530 LP=LEN(P$)
8540 PP$=LEFT$(P$,LP): P$="": GOTO 8590
8550 C$=MID$(P$,L,1): IF C$=" " THEN 8580
8560 L=L-1: IF L>0 THEN 8550
8570 L=LWID
8580 PP$=LEFT$(P$,L): P$=RIGHT$(P$,LEN(P$)-L)
```

```
5120 GOSUB 5290
5130 FOR I=1 TO LN
5140 LINEINPUT#1, L$(I)
5170 NEXT I
5180 CLOSE
5200 GOTO 5250
5210 PRINT@ 0, "START RECORDER AND PRESS ENTER.";
5215 IF INKEY$<>CHR$(13) THEN 5215
5220 INPUT#-1, L$(0): GOSUB 5290
5230 FOR I=1 TO LN
5235 INPUT#-1, L$(I)
5240 NEXT I
5250 GOSUB 3000: RETURN
5290 L$=L$(0): L=LEN(L$)
5300 CHAR= VAL(RIGHT$(L$,2)): L$=LEFT$(L$,L-2)
5310 SL= VAL(RIGHT$(L$,2)): L$=LEFT$(L$,L-5)
5320 LN=VAL(L$)
5330 RETURN
```

Subroutine to print the text file in memory to a printer.

```
6999 ' PRINTOUT RTN
7000 CLS: LIN=0
7010 X$="10": INPUT"LEFT MARGIN (DEFAULT = 10) ";X$: LM=VAL(X$)
7020 PRINT: X$="70": INPUT"RIGHT MARGIN (DEFAULT = 70) ";X$: RM=
VAL(X$)
7030 PRINT: X$="2": INPUT"LINE SPACING (DEFAULT = 2) ";X$: LS=VA
L(X$)
7040 LL=RM-LM
7050 PRINT: X$="N": INPUT"CONVERT TO LOWERCASE, UNLESS PRECEDED
BY @ (DEFAULT = NO) ";X$: UC=-1: IF LEFT$(X$,1)="Y" THEN UC=0
7070 CLS: LPRINT"": P$="": CR=0: I=0
7080 I=I+1: P$=P$+L$(I)
7090 IF RIGHT$(P$,1)=CR$ THEN CR=-1: GOTO 7110
7100 IF LEN(P$)<255-LWID AND I<LN THEN 7080
7110 GOSUB 7500: CR=0
7120 IF I<LN THEN 7080
7130 LPRINT TAB(LM);: IF UC THEN LPRINT P$;: GOTO 7150
7140 PP$=P$: GOSUB 60
7150 LPRINT""
7160 GOSUB 3000
7170 RETURN
7500 L=LL
7510 IF LEN(P$)>LL THEN 7550
7520 IF NOT CR THEN 7640
```

```
4020 X$=INKEY$: IF X$="" THEN 4020
4040 IF X$=CHR$(CLR) THEN 4220
4050 L$(0)=STR$(LN) +STR$(10000+SL*100+CHAR)
4055 PRINT@ 0, STRING$(35,32);
4060 IF X$="T" THEN 4190
4070 IF X$<>"D" THEN 4000
4075 F1$=F$
4080 PRINT@ 0, "";: LINEINPUT "FILE NAME: ";F$
4085 IF F$="" AND F1$="" THEN 4060
4088 IF F$="" THEN F$=F1$
4090 PRINT@ 0, "INSERT DISK AND PRESS ENTER.": PRINT STRING$(63,
"-");
4095 IF INKEY$<>CHR$(13) THEN 4095
4100 OPEN"O",1,F$
4140 FOR I=0 TO LN
4150 PRINT#1, L$(I)
4160 NEXT I
4170 CLOSE
4180 GOTO 4220
4190 PRINT@ 0, "START RECORDER AND PRESS ENTER.";
4195 IF INKEY$<>CHR$(13) THEN 4195
4200 FOR I=0 TO LN
4205 PRINT#-1, CHR$(34);L$(I);CHR$(34)
4210 NEXT I
4220 P=P-1: RETURN
```

Subroutine to load text from tape or disk.

```
4999 ' LOAD FROM DISK
5000 PRINT@64, STRING$(63,"-");: PRINT@ 0, STRING$(63,32);: PRIN
T@ 0, "LOAD FROM TAPE OR DISK? (T/D/CLEAR) ";
5020 X$=INKEY$: IF X$="" THEN 5020
5040 IF X$=CHR$(CLR) THEN 5250
5045 PRINT@ 0, STRING$(40,32);
5050 IF X$="T" THEN 5210
5060 IF X$<>"D" THEN 5000
5065 F1$=F$
5070 PRINT@ 0,"";: LINEINPUT "FILE NAME: ";F$
5075 IF F$="" AND F1$="" THEN 5070
5078 IF F$="" THEN F$=F1$
5080 PRINT@ 0,"INSERT DISK AND PRESS ENTER.": PRINT STRING$(63,"
-");
5085 IF INKEY$<>CHR$(13) THEN 5085
5090 OPEN"I",1,F$
5110 INPUT#1, L$(0)
```

```
8590 L$(LIN)=PP$
8600 LIN=LIN+1
8610 IF LEN(P$)>LWID THEN L=LWID: GOTO 8550
8620 IF CR AND LEN(P$)>0 THEN 8530
8630 RETURN
```

Subroutine to edit lines of text.

```
8999 ' EDIT LN SUBRTN
9000 IT=1: IF I>13 THEN V1=15: GOTO 9040
9010 V1=I+2: PRINT @PP,;
9020 X=13: IF X>LN-1 THEN X=LN-1
9030 FOR I=IT+1 TO X: PRINT L$(I): NEXT I
9040 EL=V1-(IT>13)*(IT-13)-2
9050 Q$=" UP/DOWN ARROWS:MOVE     ENTER:EDIT     D,X:DELETE
CLR:EXIT"
9060 PRINT @0,C$: PRINT STRING$(63,"-");
9080 PRINT @V1*64-2,"<E";
9082 IF PEEK(14400)>0 THEN 9088
9085 X$=INKEY$: IF X$="" THEN 9085
9088 PRINT B$;B$;
9090 IF X$<>CHR$(91) THEN 9130
9100 IF V1>3 THEN V1=V1-1: EL=EL-1: GOTO 9080
9110 IF EL=1 THEN 9080
9115 EL=EL-5: IF EL<1 THEN EL=1
9120 PRINT @127,"": FOR I=EL TO EL+12: PRINT L$(I): NEXT: GOTO 9
080
9130 IF X$<>CHR$(10) THEN 9180
9140 IF EL>=LN-1 THEN 9080
9150 EL=EL+1
9160 IF V1<15 THEN V1=V1+1: GOTO 9080
9165 NN=4: IF NN>LN-EL-1 THEN NN=LN-EL-1
9170 EL=EL+NN: PRINT @960,;: FOR I=EL-NN TO EL: PRINT L$(I): NEX
T I: GOTO 9060
9180 IF X$=CHR$(CLR) THEN 9590
9190 IF X$<>"D" OR V1=3 THEN 9250
9200 FOR J=EL TO LN-1: L$(J)=L$(J+1): NEXT: L$(LN)=""
9210 X=15-V1: IF X>LN-EL THEN X=LN-EL
9220 PRINT @V1*64-65,"": FOR J=EL TO EL+X: PRINT L$(J): NEXT
9230 IF EL=LN-1 THEN V1=V1-1: EL=EL-1
9240 LN=LN-1: GOTO 9080
9250 IF X$<>"X" THEN 9310
9260 PRINT @0, "DO YOU WANT TO DELETE FROM HERE TO THE END OF TH
E TEXT? (Y/N)  ";
9262 PRINT@V1*64-2,"<E";
```

```
9265 X$=INKEY$: IF X$="" THEN PRINT@V1%64-2,"  ";:GOTO 9262
9268 IF X$<>"Y" THEN 9060
9270 FOR J=EL TO LN: L$(J)="": NEXT J: LN=EL: CHAR=1: SL=0
9280 PRINT @V1%64-65,"": FOR J=V1 TO 14: PRINT: NEXT J
9290 PRINT @0, 0$;
9300 PRINT @V1%64-66,"<E";: GOTO 9100
9310 IF X$<>CHR$(RTN) THEN 9080
9320 L1=EL-4: IF L1<1 THEN L1=1
9330 L2=EL+4: IF L2>LN THEN L2=LN
9340 CLS: PRINT"TYPE NEW LINE BELOW (CLR-F TO FINISH)": PRINT ST
RING$(63,"-")
9350 FOR J=L1 TO EL: PRINT L$(J): NEXT J
9360 PRINT: PRINT: PRINT: PRINT
9370 FOR J=EL+1 TO L2: PRINT L$(J): NEXT J
9380 PRINT @(EL-L1+2)%64,;
9390 TLN=LN: LN=LN+1
9400 FOR J=1 TO 5: L$(TLN+J)="": NEXT J
9410 C1=CHAR: S1=SL: CHAR=1: SL=0
9420 GOSUB 500
9430 CHAR=C1: SL=S1
9440 IF L$(LN)="" THEN LN=LN-1
9450 NL=LN-TLN
9460 IF NL=1 THEN 9510
9470 IF NL>0 THEN 9500
9480 FOR J=EL TO TLN: L$(J)=L$(J+1): NEXT J
9490 L$(LN)="": GOTO 9580
9500 FOR J=LN TO EL+1 STEP-1: L$(J+NL-1)=L$(J): NEXT J
9510 FOR J=0 TO NL-1: L$(EL+J)=L$(LN+J): L$(LN+J)="": NEXT J
9520 CX$=RIGHT$(L$(EL+NL-1),1)
9530 IF CX$=CR$ THEN 9580
9540 IF CX$<>" " THEN L$(EL+NL-1)=L$(EL+NL-1)+" "
9550 SS=1: L$=L$(EL+NL): LL=LEN(L$)
9560 IF MID$(L$,SS,1)<>" " AND SS<=LL THEN SS=SS+1: GOTO 9560
9570 IF LEN(L$(EL+NL-1))+SS <=LWID THEN GOSUB 8000: GOTO 9590
9580 LN=TLN+NL-1
9590 RETURN
```

Error-handling routine.

```
19999 ' ERROR-HANDLING RTN
20000 PRINT@0,STRING$(63,32);
20010 E=ERR/2+1: PRINT@0, "ERROR:  CODE";E;
20050 PRINT"; PRESS ANY KEY";
20060 IF INKEY$="" THEN 20060
20070 P=P-1: RESUME 200
```

TRS-80® SWAT TABLE FOR:
MICROTEXT 1.2

LINES	SWAT CODE	LENGTH
10 - 70	BA	390
72 - 150	IA	183
180 - 640	TU	404
719 - 900	OM	401
920 - 1150	FH	264
1160 - 2600	NK	394
2800 - 3120	WL	345
3130 - 4020	ZD	297
4040 - 4100	YN	308
4140 - 4999	OG	209
5000 - 5085	GL	364
5090 - 5230	IH	202
5235 - 7020	ND	299
7030 - 7150	MS	396
7160 - 7590	JZ	281
7600 - 8040	UL	316
8050 - 8140	SL	313
8150 - 8600	TA	237
8610 - 9080	OQ	349
9082 - 9160	ZB	305
9165 - 9262	WS	445
9265 - 9360	SW	386
9370 - 9480	OJ	282
9490 - 19999	DB	391
20000 - 20070	KL	114

NOTES:

SWAT

by Jon R. Voskuil

TRS-80 version by Alan J. Zett

***SWAT (Strategic Weapon Against Typos)* is a debugging utility for any TRS-80.**

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. Cassette users, save *SWAT* on tape. Disk users, save the program in the ASCII format with the command SAVE"SWAT",A

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 SAVE it on tape or disk. Disk users, put a disk with the *SWAT* program in the disk drive, and type MERGE"SWAT". Tape users, after you've typed and saved the program to be tested, follow these steps:

```
A=PEEK(16562)*256+254:A=A+65536*(A>32767
):POKEA,PEEK(16548):POKEA+1,PEEK(16549):
A=PEEK(16633)+PEEK(16634)*256-2:B=INT(A/
256):A=A-B*256:POKE16548,A:POKE16549,B:C
LEAR

CLOAD (the SWAT program)

A=PEEK(16562)*256+254:A=A+65536*(A>32767
):POKE16548,PEEK(A):POKE16549,PEEK(A+1):
CLEAR
```

Once you have appended *SWAT* to your program, type RUN 65000. 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 detailed 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 saving it on tape or disk, add the following line:

```
65200 REM
```

Then, change the value of LN in line 60000 to 65200. 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        TRS-80 BASIC         SS
SS           'SWAT'            SS
SS    Author: Jon 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
```

```
65000 CLEAR99:LN=65000:NU=12:B=500
65005 CLS:MP=0:PRINT"NORMAL PARAMETERS?":GOSUB65130:IFX$<>"Y"THE
NPRINT"INPUT # OF LINES, # OF BYTES: ";:INPUTNU,B:MP=-1
65010 A=PEEK(16548)+PEEK(16549)*256:N=0:P1=PEEK(16414):P2=PEEK(1
6415):CLS:PRINT@512,STRING$(63,143);:PRINT@528," DO YOU WANT A P
RINTOUT? (Y/N) ";:GOSUB65130:CLS
65015 IFX$="Y"THENPRINT:INPUT"PROGRAM TITLE: ";T$:POKE16414,PEEK
(16422):POKE16415,PEEK(16423):PRINTCHR$(15):PRINTTAB(13)"TRS-80
SWAT TABLE FOR: ";T$:PRINT
65018 IF MP THEN PRINTTAB(13)"(MODIFIED PARAMETERS: NU="NU", B="
B")":PRINT
65020 A$="##### - #####       %%       ####":PRINTTAB(17)"LINES
       SWAT CODE   LENGTH":PRINTTAB(13)"-------------   --------
-   ------"
65030 L1=PEEK((A+2)+65536*(A+2>32767))+PEEK((A+3)+65536*(A+3>327
67))*256:A1=A:S=0:L=L1:IFL1=LNTHEN65120
65040 FORI=1TONU:AA=A:A=PEEK((A)+65536*(A>32767))+PEEK((A+1)+655
36*(A+1>32767))*256
65050 L2=L:L=PEEK((A+2)+65536*(A+2>32767))+PEEK((A+3)+65536*(A+3
>32767))*256
65060 FORJ=AA+2TOA-1:S=S+PEEK(J+65536*(J>32767)):NEXT
65070 IFL=LNORA-A1>BTHENI=NU
65080 NEXT
65090 D=INT(S/676):S=S-D*676:D1=INT(S/26):D2=S-D1*26:C$=CHR$(D1+
65)+CHR$(D2+65)
65100 PRINTTAB(13);USINGA$;L1;L2;C$;A-A1:IFX$<>"Y"THENN=N+1:IFN=
14THENN=0:PRINT:PRINT@960,STRING$(63,143);:PRINT@978," PRESS <EN
TER> TO CONTINUE ";:GOSUB65130:PRINT@896,CHR$(31);
65110 GOTO65030
65120 PRINT:POKE16414,P1:POKE16415,P2:PRINT:END
65130 X$=INKEY$:IFX$=""THEN65130ELSERETURN
```

TRS-80® SWAT TABLE FOR:
SWAT

LINES	SWAT CODE	LENGTH
65000 - 65020	FL	542
65030 - 65100	IH	502
65110 - 65130	WI	66

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