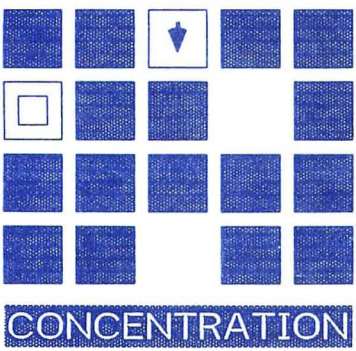
SoftSide®

FRONT RUNNER

Printed Game Software for Atari® , Apple® & IBM® PC.

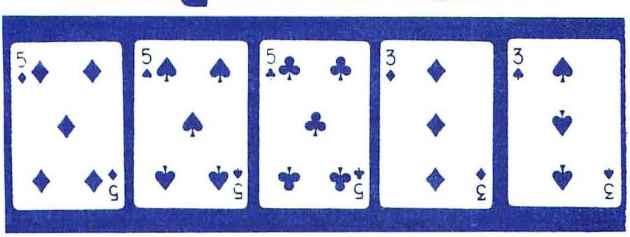


CONCENTRATION

#47

AT THE CODFISH BALL

CRIBBAGE SQUARES

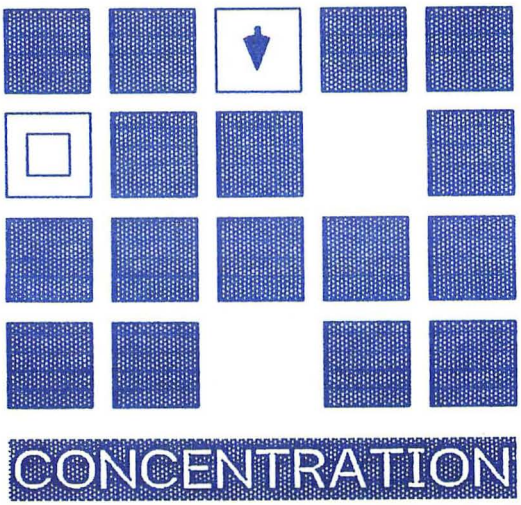


A product of **SoftSide** Publications, Inc.
10 Northern Boulevard, Amherst, NH 03031

SoftSide®

FRONT RUNNER

Index



by Glenn Archer
Translation by Kerry Shetline

Concentration is a game program for an IBM® PC with disk drive, Color Graphics Adapter and 64K RAM.

"Let's play *Concentration*!" Remember the television game show? Now, you can play this exciting game on your IBM PC. You also can challenge and hone your memory since this game, even in Level 1, is more difficult than either the television or the board version.

The object of *Concentration* is to match all of the shapes on the board. The winner (in a two player game) is the player who matches the most shapes.

The game first asks the number of players. In the one-player game, the object is to match all the shapes in the least number of moves. This also applies to the two-player game, but the players are trying to get a better score as well.

After receiving the players' names, the computer asks for the difficulty level. Of the three levels, Level 1 is the easiest, with a small board and easy shapes to remember; Level 2 is a slightly larger board, with harder shapes; and Level 3 is the largest board, with shapes that look almost the same.

After you choose the difficulty level, the computer randomly mixes and matches the shapes. It then draws the board, numbers and letters on the screen. These numbers and letters are the coordinates you enter for your guesses. Enter the row (numbers) first, and follow with the column (letters). You need not press RETURN after entering the letter. The shape in the corresponding square appears and you enter the second shape. If you enter the same coordinate as your first guess, or the square entered has already been matched, it rejects the input and you must reenter your guess.

If the two shapes don't match, they are displayed a little longer and then covered up. Then you guess again or, in case of a two player game, the next player guesses. If the two shapes match, the player is awarded one point, and the two shapes are cleared from the screen.

The game continues until all shapes have been matched. It then prints the winner (two player game) and the number of moves it took to complete the game.

Variables

A$, B$, PP$: Input variables.
GR%(*): Graphics array.
H$: Contains letter indicating rightmost column to input.
J, J1, K, Q: Loop variables.
L: Used in shape mixing routine.
LM: Number of columns on game board.
LV: Level of difficulty.
MV: Number of moves player(s) have taken.
N: Number of shapes matched.
N$(2): Array containing players' names.
N1: Shape number of player's first guess.
N2: Shape number of player's second guess.
NU: Shape number of player's guess.
PP: Number of players.
SC(2): Scores of player(s).
SH(72): Array containing the number of each square's shape.
ST(72): Flag array to test if a shape has already been matched.
TG: Number of shapes to match.
X, Y: X and Y coordinates of shape guessed.
X1, Y1: X and Y coordinates of first shape guessed.
X2, Y2: X and Y coordinates of second shape guessed.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                           SS
SS     PC Advanced BASIC     SS
SS      'Concentration'      SS
SS    Author: Glenn Archer   SS
SS Translator: Kerry Shetline SS
SS     Copyright © 1983      SS
SS SoftSide Publications, Inc SS
SS                           SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on Issue #47 SoftSide DV.

Initialization.

```
100 SCREEN 1,0:CLS:DIM SH(72),ST(72),GR%(9):DEF FNUP$(X$)=CHR$(ASC(X$)+32*(ASC(X$)>96)):BL$=STRING$(40,32):L4$=STRING$(159,32):BS$=CHR$(29):CSR$=CHR$(22)+BS$:RANDOMIZE VAL(MID$(TIME$,3,2))*100+VAL(RIGHT$(TIME$,2))
110 LOCATE ,13:PRINT "CONCENTRATION!":PRINT:PRINT TAB(19)"by":PRINT:PRINT TAB(14)"Glenn Archer"
```

Get the number of players.

```
120 LOCATE 9:PRINT "How many players (1 or 2) ? "CSR$;
130 PP$=INPUT$(1):IF PP$<"1" OR PP$>"2" THEN 130
140 PRINT PP$:PP=VAL(PP$)
```

Get the players' names.

```
150 FOR J=1 TO PP
160 LOCATE 10+(2*J):PRINT "What is player #"J;:LINE INPUT "'s name? ";NM$:IF LEN(NM$)<1 THEN 160
170 IF LEN(NM$)<8 THEN GOSUB 1150:N$(J)=NM$:NEXT:GOTO 190
180 LOCATE 16:PRINT "Please enter a maximum of 7 letters only":FOR K=1 TO 2000:NEXT:LOCATE 16:PRINT BL$;:LOCATE 10+(2*J):PRINT BL$;:GOTO 160
```

Get the difficulty level.

```
190 PRINT:PRINT:PRINT "Level of difficulty (1-3) ? "CSR$;
200 A$=INPUT$(1):IF A$<"1" OR A$>"3" THEN 200
```

```
210 PRINT A$:LV=VAL(A$)
```

Create random layout of shapes.

```
220 CLS:LOCATE 10,9:PRINT "Generating bo
ard pieces"
230 LOCATE 12,15:PRINT "Please wait"
240 ON LV GOSUB 1080,1090,1100
250 ON LV GOSUB 1080,1090,1100
```

Set up screen display.

```
260 CLS:LINE (0,0)-(279,159),,B:LINE (19
,19)-(159-60*(LV=2)-98*(LV=3),139),,BF
270 FOR J=19 TO 259 STEP 20:LINE (J,19)-
(J+1,139),0,B:NEXT:FOR J=19 TO 139 STEP
20:LINE (19,J)-(259,J+1),0,B:NEXT
280 FOR X=49 TO 54:LOCATE 1,39:PRINT CHR
$(X);:GET (304,0)-(311,7),GR%:PRINT BS$"
 ":PUT (10,27+20*(X-49)),GR%: NEXT

290 FOR X=65 TO 71-3*(LV=2)-5*(LV=3):LOC
ATE 1,39:PRINT CHR$(X);:GET (304,0)-(311
,7),GR%:PRINT BS$" ":PUT (27+20*(X-65),1
44),GR%:NEXT
```

Main program loop for handling each player.

```
300 FOR Q=1 TO PP
310 LOCATE 22:PRINT N$(1)"'s score:"SC(1
);:IF PP=2 THEN LOCATE ,21:PRINT N$(2)"'
s score:"SC(2)
```

Get player's first guess.

```
320 LOCATE 23,1:PRINT BL$;:LOCATE 23,1:P
RINT N$(Q)", enter guess #1: ";:GOSUB 44
0:N1=NU:X1=X:Y1=Y:IF ST(NU)=-1 THEN GOSU
B 580:LOCATE 23,1:PRINT BL$;:GOTO 320
330 GOSUB 530
```

Get player's second guess.

```
340 LOCATE 23:PRINT BL$;:LOCATE 23:PRINT
 N$(Q)", enter guess #2: ";:GOSUB 440:IF
 N1=NU THEN 340
350 N2=NU:X2=X:Y2=Y:IF ST(NU)=-1 THEN GO
SUB 580:GOTO 340
360 GOSUB 530
```

Check if shapes match.

```
370 IF SH(N1)=SH(N2) THEN 400
```

Shapes did not match.

```
380 LOCATE 21:PRINT L4$;:LOCATE 23,1:PRI
NT TAB(12)"Sorry, no match.":FOR J=1 TO
3000:NEXT:GOSUB 560
390 MV=MV+1:NEXT Q:GOTO 300
```

Player has matched shapes.

```
400 LOCATE 21:PRINT L4$;:LOCATE 23,1:FOR
 K=700 TO 1200 STEP 100:SOUND K,2:NEXT:L
OCATE ,2:PRINT "Fantastic, "N$(Q)"! You
matched them!":FOR J=1 TO 3000:NEXT
```

Increase player's score, flag matched shapes, increment move counter and black out shapes.

```
410 SC(Q)=SC(Q)+1:ST(N1)=-1:ST(N2)=-1:MV
=MV+1:Y=Y1:X=X1:GOSUB 570:Y=Y2:X=X2:GOSU
B 570
420 N=N+1:IF N=TG THEN 950
430 GOTO 310
```

Get a guess.

```
440 PRINT CSR$;:A$=FNUP$(INPUT$(1));:IF A
SC(A$)=6 THEN 1070
450 IF A$<"1" OR A$>"6" THEN 440
460 PRINT A$;
470 PRINT CSR$;:B$=FNUP$(INPUT$(1));:IF A
SC(B$)=6 THEN 1070
480 IF ASC(B$)=8 THEN PRINT " "BS$;BS$;:
GOTO 440
490 IF B$<"A" OR B$>H$ THEN 470
500 PRINT B$:IF A$="1" THEN NU=ASC(B$)-6
4:GOTO 520
510 NU=((VAL(A$)-1)*LM)+(ASC(B$)-64)
520 X=(ASC(B$)-64)*20:Y=VAL(A$)*20:RETUR
N
```

Draw routine.

```
530 GOSUB 570
540 FOR J=400 TO 700 STEP 50:SOUND J,.1:
NEXT
550 ON SH(NU) GOTO 590,600,610,620,630,6
40,650,660,670,680,690,700,710,720,730,7
40,750,760,770,780,790,800,810,820,830,8
40,850,860,870,880,890,900,910,920,930,9
40
```

Fill in squares after an incorrect guess.

```
560 LINE (X1+1,Y1+1)-(X1+18,Y1+18),,BF:L
INE (X2+1,Y2+1)-(X2+18,Y2+18),,BF:RETURN
```

Black out a square.

```
570 LINE (X,Y)-(X+19,Y+19),0,BF:RETURN
```

Tell player that a square has already been matched.

```
580 LOCATE 21,1:PRINT L4$;:LOCATE 23,1:P
RINT "That square has already been match
ed.":FOR J=1 TO 3000:NEXT:RETURN
```

Routines for drawing various shapes.

```
590 LINE (X+2,Y+2)-(X+17,Y+2):LINE -(X+1
7,Y+17):LINE -(X+2,Y+17): LINE -(X+2,Y+2
):RETURN
600 LINE (X+9,Y+2)-(X+17,Y+17):LINE -(X+
2,Y+17):LINE -(X+9,Y+2):RETURN
610 LINE (X+2,Y+2)-(X+17,Y+2):LINE -(X+9
,Y+17):LINE -(X+2,Y+2):RETURN
620 LINE (X+2,Y+2)-(X+17,Y+17):LINE (X+1
7,Y+2)-(X+2,Y+17):RETURN
630 LINE (X+9,Y+2)-(X+9,Y+16):LINE (X+2,
Y+9)-(X+16,Y+9):RETURN
640 LINE (X+9,Y+2)-(X+16,Y+9):LINE -(X+9
,Y+16):LINE -(X+2,Y+9):LINE -(X+9,Y+2):R
ETURN
650 LINE (X+6,Y+6)-(X+13,Y+6):LINE -(X+1
3,Y+13):LINE -(X+6,Y+13):LINE -(X+6,Y+6)
:RETURN
660 LINE (X+9,Y+2)-(X+9,Y+17):LINE (X+4,
Y+7)-(X+9,Y+2):LINE -(X+14,Y+7):RETURN
670 LINE (X+2,Y+9)-(X+17,Y+9):LINE (X+7,
Y+4)-(X+2,Y+9):LINE -(X+7,Y+14):RETURN
680 LINE (X+2,Y+9)-(X+17,Y+9):LINE (X+12
,Y+4)-(X+17,Y+9):LINE -(X+12,Y+14):RETUR
N
690 LINE (X+9,Y+2)-(X+9,Y+17):LINE (X+4,
Y+12)-(X+9,Y+17):LINE -(X+14,Y+12):RETUR
N
700 LINE (X+2,Y+2)-(X+17,Y+17):RETURN
710 LINE (X+17,Y+2)-(X+2,Y+17):RETURN
720 GOSUB 590:GOTO 650
730 GOSUB 640:GOTO 630
740 GOSUB 670:GOTO 680
750 GOSUB 710:GOTO 640
760 GOSUB 700:GOTO 640
770 GOSUB 630:GOTO 740
780 LINE (X+2,Y+2)-(X+13,Y+2):LINE -(X+1
3,Y+13):LINE -(X+2,Y+13):LINE -(X+2,Y+2)
:LINE (X+17,Y+17)-(X+6,Y+17):LINE -(X+6,
Y+6):LINE -(X+17,Y+6):LINE -(X+17,Y+17):
RETURN
790 LINE (X+4,Y+4)-(X+7,Y+7),,BF:LINE (X
+12,Y+12)-(X+15,Y+15),,BF:RETURN
800 GOSUB 590:GOTO 790
810 GOSUB 700:GOTO 590
820 GOSUB 710:GOTO 590
830 GOSUB 640:GOTO 650
840 GOSUB 630:GOTO 620
850 GOSUB 590:GOTO 620
860 GOSUB 590:GOTO 840
870 GOSUB 610:GOTO 600
880 GOSUB 870:GOTO 590
890 GOSUB 850:GOTO 930
900 GOSUB 620:GOTO 640
910 GOSUB 900:GOTO 590
920 GOSUB 850:GOTO 780
930 GOSUB 590:GOTO 730
940 GOSUB 660:GOTO 690
```

End of game.

```
950 CLS:LOCATE ,11:PRINT "*** Final scor
e ***":LOCATE 5
960 IF PP=1 THEN PRINT N$(1)", you score
d"SC(1)"points.":GOTO 1020
970 IF SC(1)=SC(2) THEN A$="It's a tie!"
980 IF SC(1)<SC(2) THEN A$=N$(2)+" won!"
990 IF SC(1)>SC(2) THEN A$=N$(1)+" won!"
1000 LOCATE ,(20-LEN(A$)/2):PRINT A$
1010 PRINT:PRINT:PRINT N$(1)"'s score wa
s"STR$(SC(1))".":PRINT:PRINT N$(2)"'s sc
ore was"STR$(SC(2))"."
1020 PRINT:PRINT "It took"MV"turns to ma
tch all":PRINT:PRINT "of the shapes."
1030 FOR J=1 TO 3:FOR K=600 TO 900 STEP
20:SOUND K,.2:NEXT:NEXT
```

Check if another game is to be played.

```
1040 LOCATE 22:PRINT "Do you want to pla
y again? "CSR$;
1050 A$=FNUP$(INPUT$(1)):IF A$="Y" THEN
RUN
1060 IF A$<>"N" THEN 1050
1070 CLS:END
```

Routines to set up different levels of difficulty.

```
1080 TG=21:LM=7:H$="G":GOTO 1110
1090 TG=30:LM=10:H$="J":GOTO 1110
1100 TG=36:LM=12:H$="L":GOTO 1110
```

CONCENTRATION

Mix and match shapes.

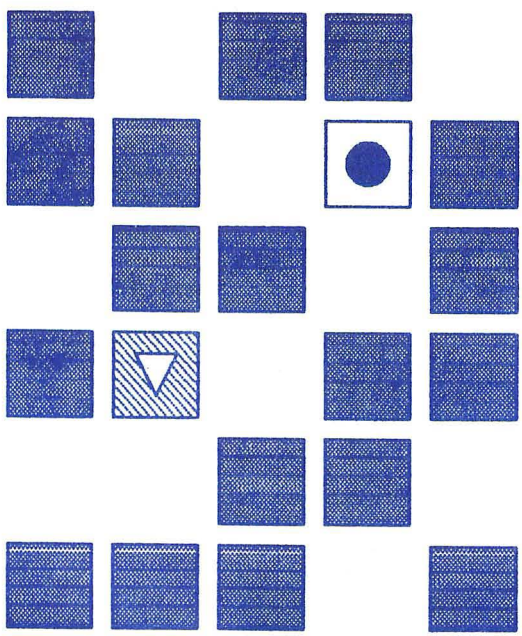
```
1110 FOR J=1 TO TG
1120 K=INT((TG*2)*RND(1)+1):IF SH(K)<>0
THEN 1120
1130 L=INT(TG*RND(1)+1):IF ST(L)<>0 THEN
 1130
1140 SH(K)=L:ST(L)=1:NEXT:FOR J=1 TO TG:
ST(J)=0:NEXT:RETURN
```

Make sure players' names start in upper case, and the rest of the letters are lower case.

```
1150 MID$(NM$,1)=FNUP$(MID$(NM$,1,1)):FO
R K=2 TO LEN(NM$):C$=MID$(NM$,K,1):MID$(
NM$,K)=CHR$(ASC(C$)-32*(ASC(C$)>64 AND A
SC(C$)<91)):NEXT:RETURN
```

STOMP TABLE

LINES	STOMP CODE	LENGTH
100 - 190	SQ	541
200 - 310	TQ	528
320 - 430	JQ	494
440 - 550	ZQ	428
560 - 650	AS	552
660 - 770	SP	360
780 - 890	GS	294
900 - 1010	QW	323
1020 - 1130	TJ	328
1140 - 1150	IC	147



AT THE CODFISH BALL

by David Plotkin

At The Codfish Ball is a game for the Atari 400, 800 1200 with 48K RAM, one disk drive and a joystick.

This game is different from other arcade style games because it does not require supernormal reflexes to achieve decent scores, but instead concentrates on skill and timing. It is also a good example of how a game can be written in BASIC and still be smooth and fast. Take a look at the "Take-Apart" for programming hints and watch *SoftSide* for more information on how to write BASIC games.

You control "Diver Dave." He moves back and forth along the bottom of the screen under control of your joystick plugged into port 1. Pressing the fire button releases a spear from Diver Dave's spear gun which moves up the screen until it hits a target or goes off the top of the screen. You can only have one spear on the screen at once.

The targets in this undersea shooting gallery are whales, sea snakes, rays, soles, seahorses, sea snails and, of course, cod. Each hit on a target nets you ten points. The program keeps track of score, high score and number of remaining spears. Each time you clear the screen, the action speeds up and the number of spears you start with decreases. The game is over when all the spears are gone.

In line 20010 of the listing there is a string variable XFR$. XFR$ contains a machine language routine that quickly sets up a redefined character set for the Atari. It is very difficult to represent this string in the listing of the program, so here are explicit instructions on how to type it:

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

Type the following sequence exactly to produce XFR$:

[h] [Atari] [)] [Atari] [CTRL-,] [Atari] [CTRL-E] [K] [CTRL-E] [M] [)] [CTRL-.] [CTRL-E] [N] [%] [Atari] [j] [CTRL-X] [i] [CTRL-A] [Atari] [CTRL-E] [L] [spacebar] [Atari] [CTRL-,] [Atari] [1] [M] [CTRL-Q] [K] [H] [P] [y] [f] [L] [f] [N] [%] [N] [I] [d] [P] [m] [Atari] [CTRL-.]

Variables

HISCORE: Keeps track of high score.
FLAG: Status of spear (0, off screen; 1, on screen).
B: Level of difficulty.
SCORE: Player's score.
BULLETS: Number of spears.
YP2: Screen line of spear.
UP$: String which contains machine language routine to move the spear up.
MOVE$: String which contains machine language routine to move the targets.
PM: Address of start of Player/Missile memory.
PXP2: X coordinate of spear in P/M coordinates.
JUMP$: String which contains machine language routine to move the spear around the screen.
XP2: X coordinate of target hit.
PYP2: Y coordinate of spear.
HITS: Number of hits on targets
BYTE$,A,A1,A2,A3,A$: Variables for entering machine code routines.
Z$: Holds value of current target to put on the screen.
DL: Address of display list LMS bytes.
MEM: Address of second line of display list.
ML,MH: Low and high bytes of MEM.
PXP1: X coordinate of Diver Dave.
SC: Address of beginning of screen memory.
START: Address of alternate character set.
W,I,J,C,AA,X,LINE,COLUMN: Miscellaneous loop and read variables.

```
 SS SS SS SS SS SS SS SS SS SS SS
 SS                             SS
 SS        Atari BASIC          SS
 SS       'Codfish Ball'        SS
 SS   Author: David Plotkin     SS
 SS     Copyright © 1983        SS
 SS SoftSide Publications, Inc SS
 SS                             SS
 SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on issue #47 SoftSide DV and CV.

Do subroutine calls and other set-up steps, initialize variables.

```
10 GOSUB 10500:GOSUB 11000:GOSUB 12000
:GOSUB 13000:GOSUB 20000:HISCORE=0
20 FLAG=0:B=0:SCORE=0:BULLETS=300:GOSU
B 10000
```

Test for spear on the screen. Make some sound to slow program execution, move the targets and check to see if player wants to fire if there is no spear on the screen.

```
100 IF FLAG=0 THEN D=USR(ADR(MOVE$)):S
OUND 2,10,12,6:SOUND 2,20,12,6:SOUND 2
,30,12,6:SOUND 2,40,12,6:SOUND 2,0,0,0
:GOTO 240
```

Move the spear up.

```
110 YP2=YP2-1:D=USR(ADR(UP$),PM+656)
```

Test for spear still on screen.

```
120 IF YP2>0 THEN GOTO 160
```

Spear has reached the top of the screen. Move back to the bottom of the screen, reset the position and status of the spear.

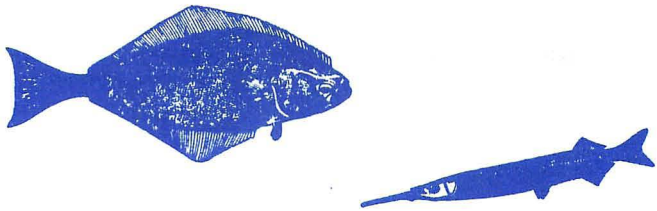
```
130 PXP2=0:POKE 53249,PXP2:D=USR(ADR(J
UMP$),PM+736,PM+656)
140 D=USR(ADR(JUMP$),PM+656,PM+640):FL
AG=0:IF BULLETS=0 THEN GOTO 500
150 XP2=0:YP2=20:POKE 53278,0:GOTO 220
```

Test to see if the spear hit a target.

```
160 IF PEEK(53253)=0 THEN GOTO 220
```

The spear hit a target. Move the spear off the screen and erase the target. Make some sound, reset the position and status of the spear, increment the score and number of hits and test to see if the screen has been cleared or the diver has run out of spears.

```
170 PYP2=YP2*4+656:PXP2=0:POKE 53249,P
XP2
180 D=USR(ADR(JUMP$),PM+736,PM+PYP2):D
=USR(ADR(JUMP$),PM+PYP2,PM+640)
```



```
190 COLOR 32:PLOT XP2,YP2:PLOT XP2,YP2
+1:PLOT XP2-1,YP2:PLOT XP2-1,YP2+1:PLO
T XP2+1,YP2:PLOT XP2+1,YP2+1
195 FOR W=1 TO 10:SOUND 0,W*10,8,8:NEX
T W:SOUND 0,0,0,0
200 XP2=0:YP2=20:FLAG=0:SCORE=SCORE+10
:POSITION 15,0:PRINT SCORE;:POKE 53278
,0
210 HITS=HITS+1:IF HITS=72 THEN GOTO 3
00
215 IF BULLETS=0 THEN GOTO 500
```

Move the targets and test for a hit. If there has been a hit, go back to line 170.

```
220 D=USR(ADR(MOVE$)):POKE 53278,0:IF
PEEK(53253)<>0 THEN GOTO 170
```

Go back through the program loop.

```
230 GOTO 100
```

Test to see if the trigger button was pushed.

```
240 IF STRIG(0)=1 THEN 100
```

Trigger button is pushed. Launch a new spear, decrement the number of spears, and update the spear status.

```
250 POKE 77,0:FLAG=1:PXP2=PEEK(1662)+4
:XP2=INT((PXP2-48)/4)
260 POKE 53249,PXP2:BULLETS=BULLETS-1:
POSITION 5,0:PRINT BULLETS;" ";:GOTO 1
10
```

Screen cleared. Increase difficulty and give the diver some new spears. Redraw the targets.

```
300 B=B+1:BULLETS=300-10*B:IF B>12 THE
N B=12
310 GOSUB 2200:HITS=0:GOSUB 10005:GOTO
 100
```

Game over. Clear the remaining targets from the screen, display the game over message and high score. Restart the game if player presses the fire button.

```
500 GOSUB 2200:POSITION 5,15:? "GAME O
VER.  TO PLAY AGAIN"
510 POSITION 5,16:? "PRESS FIRE BUTTON
":IF SCORE>HISCORE THEN HISCORE=SCORE
520 POSITION 5,17:? "HIGH SCORE SO FAR
 IS ";HISCORE
530 IF STRIG(0)=1 THEN 530
540 FLAG=0:B=0:SCORE=0:BULLETS=300:GOS
UB 2200:GOSUB 10005:GOTO 100
```

Read hex code bytes from 13045-13240 and translate into numbers.

```
2000 READ BYTE$:A1=ASC(BYTE$)-48:IF A1
>10 THEN A1=A1-7
2010 A2=ASC(BYTE$(2,2))-48:IF A2>10 TH
EN A2=A2-7
2020 A$=CHR$(A1*16+A2):RETURN
```

Break the address of each line of targets into high and low bytes.

```
2100 FOR J=0 TO 3:A2=INT(A1/256):A3=A1
-A2*256:POKE A,A2:POKE A+1,A3
2120 A=A+3:A1=A1+80:NEXT J:RETURN
```

The clear screen routine.

```
2200 COLOR 32:FOR J=1 TO 23:PLOT 1,J:D
RAWTO 39,J:NEXT J:POSITION 0,0:RETURN
```

Puts the targets on the screen.

```
10000 DIM Z$(1)
10005 FOR LINE=2 TO 16 STEP 2:Z$=CHR$(
LINE/2-1):FOR COLUMN=3 TO 36 STEP 4
10010 POSITION COLUMN,LINE:PRINT Z$
10020 NEXT COLUMN:NEXT LINE:RESTORE 10
100+B:FOR J=1 TO 8:READ A:POKE 1663+J*
3,A:NEXT J
```

```
10030 POSITION 0,0:? "AMMO    +++";BUL
LETS;" SCORE      +++++";SCORE:REM + S
IGNS ARE ESC CTRL-RIGHT ARROW
10050 RETURN
10100 DATA 0,2,0,2,0,2,0,2
10101 DATA 0,2,0,2,0,2,1,2
10102 DATA 0,2,0,2,1,2,1,2
10103 DATA 0,2,1,2,1,2,1,2
10104 DATA 1,2,1,2,1,2,1,2
10105 DATA 0,2,0,2,0,2,0,3
10106 DATA 0,2,0,2,0,3,0,3
10107 DATA 0,2,0,3,0,3,0,3
10108 DATA 0,3,0,3,0,3,0,3
10109 DATA 0,3,0,3,0,3,1,3
10110 DATA 0,3,0,3,1,3,1,3
10111 DATA 0,3,1,3,1,3,1,3
10112 DATA 1,3,1,3,1,3,1,3
```

Routine for Diver movement on immediate VBI.

```
10500 RESTORE 10510:FOR I=1536 TO 1611
:READ A:POKE I,A:NEXT I
10510 DATA 104,173,34,2,141,74
10520 DATA 6,173,35,2,141,75
10530 DATA 6,169,6,162,6,160
10540 DATA 23,32,92,228,96,24
10550 DATA 173,126,6,141,0,208
10560 DATA 173,124,2,208,6,206
10570 DATA 126,6,206,126,6,173
10580 DATA 125,2,208,6,238,126
10590 DATA 6,238,126,6,173,126
10600 DATA 6,201,48,176,5,169
10610 DATA 192,141,126,6,201,193
10620 DATA 144,5,169,50,141,126
10630 DATA 6,76,73,6
10640 IF PEEK(547)<>6 THEN DUM=USR(153
6)
10650 RETURN
```

Modifies the display list and pokes in the colors for each band of color.

```
11000 GRAPHICS 21:POKE 752,1:POKE 87,0
:C=0:RESTORE 11160
11010 DL=PEEK(560)+256*PEEK(561)+4
11020 MEM=PEEK(DL)+256*PEEK(DL+1)+40
11030 MH=INT(MEM/256):ML=MEM-MH*256
11040 POKE DL-1,198:POKE DL+2,66:POKE
DL+3,ML:POKE DL+4,MH
11050 FOR J=5 TO 26:POKE DL+J,2:NEXT J
11060 FOR J=5 TO 19 STEP 2:POKE DL+J,1
30:NEXT J
11070 POKE DL+27,65
11080 POKE DL+28,PEEK(560)
11090 POKE DL+29,PEEK(561)
11100 READ A:IF A=999 THEN 11120
11110 POKE 1744+C,A:C=C+1:GOTO 11100
11120 POKE 512,208:POKE 513,6
11130 POKE 1774,98:POKE 1775,114:POKE
1776,130:POKE 1777,146:POKE 1778,162:P
OKE 1779,178:POKE 1780,194
11140 POKE 1781,98:POKE 1782,114
11150 POKE 1783,0:POKE 54286,192
11160 DATA 72,138,72,174,247,6,189,238
,6,141,10,212,141,24,208,232
11170 DATA 224,9,144,2,162,0,138,141
11180 DATA 247,6,104,170,104,64,999
11190 POSITION 0,0:PRINT "INITIALIZING
":RETURN
```

Set up Player/Missile graphics and place shapes in P/M memory.

```
12000 A=PEEK(106)-8:POKE 54279,A:PM=25
6*A:POKE 106,A:POKE 623,0
12010 POKE 559,46:POKE 53277,3:POKE 70
4,8:POKE 705,0
12020 FOR I=PM+512 TO PM+768:POKE I,0:
NEXT I
12030 FOR J=PM+612 TO PM+620:READ A:PO
KE J,A:NEXT J
12040 DATA 60,102,102,60,153,126,24,24
,60
12050 PXP1=100:POKE 53248,PXP1:POKE 16
62,PXP1
12060 FOR J=PM+736 TO PM+739:READ A:PO
KE J,A:NEXT J
12070 DATA 192,192,192,192
12080 PXP2=0:POKE 53249,PXP2:YP2=20
12090 RETURN
```

Initializes the routines to move the targets and spears.

```
13000 DIM MOVE$(200),BYTE$(2),A$(40),U
P$(28),DOWN$(28),JUMP$(26):FOR J=1 TO
171:GOSUB 2000:MOVE$(J,J)=A$:NEXT J
13010 SC=PEEK(88)+PEEK(89)*256:A=1664
13020 A1=SC+2+80:GOSUB 2100
13030 A1=SC+2+40*10:GOSUB 2100
13040 POKE A,0
13045 DATA D8,A9,80,85,CD,A9,06,85,CE
```

```
13050 DATA 38,B0,3D,60,A0,00,B1,CB
13055 DATA 48,A2,25,C8,B1,CB,88,91
13060 DATA CB,C8,CA,D0,F6,68,A0,25
13065 DATA 91,CB,38,B0,27,A0,00,B1
13070 DATA CB,48,C8,B1,CB,48,A2,24
13075 DATA C8,B1,CB,88,88,91,CB,C8
13080 DATA C8,CA,D0,F4,68,A0,25,91
13085 DATA CB,68,88,91,CB,38,B0,04
13090 DATA D8,68,C6,CD,E6,CD,A0,00
13095 DATA B1,CD,F0,B7,85,CC,E6,CD
13100 DATA B1,CD,85,CB,E6,CD,B1,CD,F0,
0C
13105 DATA C9,01,EA,F0,20,C9,02,EA
13110 DATA F0,A0,D0,B7,A0,25,B1,CB
13115 DATA 48,A2,25,88,B1,CB,C8,91
13120 DATA CB,88,CA,D0,F6,68,A0,00
13125 DATA 91,CB,38,B0,C5,A0,25,B1
13130 DATA CB,48,88,B1,CB,48,A2,24
13140 DATA 88,B1,CB,C8,C8,91,CB,88,88,
CA
13145 DATA D0,F4,68,A0,00,91,CB,68
13150 DATA C8,91,CB,38,B0,A2
13200 FOR J=1 TO 28:GOSUB 2000:UP$(J,J
)=A$:NEXT J
13210 DATA 68,68,85,CC,68,85,CB,A2
13220 DATA 60,A0,00,C8,C8,C8,C8,B1
13230 DATA CB,88,88,88,88,91,CB,C8
13240 DATA CA,D0,F0,60
13300 DOWN$=UP$:DOWN$(11,11)=CHR$(94):
FOR J=12 TO 15:DOWN$(J,J)=CHR$(136):NE
XT J
13310 FOR J=18 TO 21:DOWN$(J,J)=CHR$(2
00):NEXT J:DOWN$(24,24)=CHR$(136)
13320 FOR J=1 TO 25:READ AA:JUMP$(J,J)
=CHR$(AA):NEXT J:RETURN
13330 DATA 104,104,133,204,104,133,203
,104,133,207,104,133,206,160,0,177,206
,145,203,200,192,8,208,247,96
```

Downloads and modifies the alternate character set.

```
20000 POKE 106,PEEK(106)-5:START=(PEEK
(106)+1)*256:POKE 756,START/256:POKE 7
52,1
20010 DIM XFR$(38):XFR$="h)KM)'N%j
iL1MKHPyfLfN%NIdPm'":REM SEE DOCU
MENTATION
20020 Z=USR(ADR(XFR$)):RESTORE 20050
20030 READ X:IF X=-1 THEN RESTORE :RET
URN
20040 FOR Y=0 TO 7:READ Z:POKE X+Y+STA
RT,Z:NEXT Y:GOTO 20030
20050 DATA 512,27,14,38,118,254,254,12
0,255
20060 DATA 520,80,32,32,32,115,251,223
,255
20070 DATA 528,0,4,103,148,146,146,146
,140
20080 DATA 536,255,246,252,184,248,228
,194,129
20090 DATA 544,224,188,192,198,201,201
,199,126
20100 DATA 552,248,160,230,152,24,103,
4,4
20110 DATA 560,0,0,12,222,123,222,12,0
20120 DATA 568,8,24,56,254,19,2,0,0
20130 DATA -1
```

For ATARI® CODFISH BALL

LINES	STOMP CODE	LENGTH
10 - 170	HJ	501
180 - 260	ZX	568
300 - 2100	PF	515
2120 - 10103	LH	452
10104 - 10520	KA	323
10530 - 10640	QN	333
10650 - 11100	XT	380
11110 - 12020	DQ	529
12030 - 13040	LH	442
13045 - 13100	BJ	393
13105 - 13220	CB	395
13230 - 20020	HK	522
20030 - 20130	PW	415

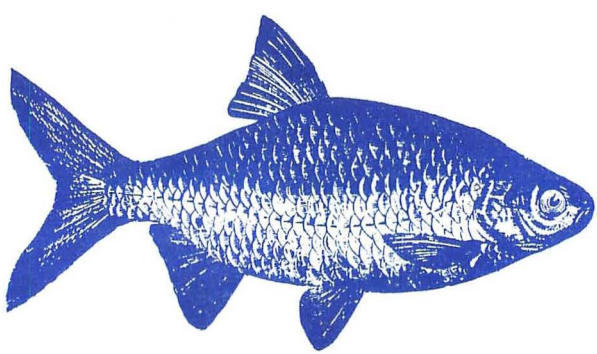
TAKE-APART: At the Codfish Ball

"At the Codfish Ball" is a BASIC and machine language game which runs fairly smoothly and with appreciable speed. Two factors combine to produce this satisfactory result, and we're going to look at both of them.

At the heart of "Codfish Ball" are the machine language routines set up by some of the subroutine calls in lines 10 and 20. These routines handle all the motion in the program. Before analyzing how the machine language routines work and how they impact the program, I should point out that I did not author a single one of the routines. They have appeared in various periodicals over the last two years. While I did need to modify the routines somewhat to conform to the requirements of this program, the fact that I didn't write the routines points up an important fact: You don't need to be an assembly language expert (I'm not!) to write good, fun-to-play programs on your Atari. In fact, in many cases you need no assembly language at all because the routines you need already exist. You just have to copy them into your program.

I'll talk about the machine language subroutines in the order that they appear in the program. The first routine begins at line 10500 and ends at line 10650. This utility reads joystick 1 and moves player 1 (the diver) left and right across the bottom of the screen. The routine is located on page six of memory beginning at memory location 1536. While you may know that POKEing the horizontal position register will move a Player or Missile horizontally very quickly, the process of reading the joystick, computing the new value of the horizontal position, and placing it in the horizontal position register can slow down a program considerably. Not only does this routine execute the whole sequence at machine speed, even the necessity of a USR call to the routine has been eliminated by placing the code execution on the Immediate Vertical Blank Interrupt. Without getting technical, this means that the code gets executed 60 times per second. Thus, there will never be a delay between the time you move the joystick and the response on the screen.

The subroutine from 11000-11190 is the modified display list. Except for a minor modification, this display list is identical to the one used for "Hopper." For information and a marvelously informative take-apart, see "Exploring the Atari Frontier" by Alan J. Zett in January, 1983 ***SoftSide.*** The modification puts more bands of color on the screen (nine, excluding the Graphics 1 line at the top) than there were in "Hopper." The second byte of data in line 11170 was changed from a four to a nine, and more lines of display list were POKEd



with the appropriate value (130) to activate the Display List Interrupt in line 11060. The nine colors are stored in page six locations 1774-1782 and placed in those registers in lines 11130 and 11140. If you don't like the colors, you can change them by poking a different number into these page six locations (e.g., POKE 1774,0 to place a black band on the screen).

The most powerful of the machine language routines in this program is the next one, from line 13000 to 13150. It should look familiar — it is the routine used to move the cars, logs, etc. in "Hopper." Lines 13010 to 13030 place the address of each line (up to eight total) that will scroll into the locations on page six that are used by the program. The start of screen memory is computed as:

SC = PEEK(88) + 256*PEEK(89)

Then an offset is added to SC to move down some lines and get away from the left edge of the screen in line 13020. The addition of two bytes to SC moves the address two spaces from the left edge of the screen, and the addition of 80 bytes moves the address down two lines from the top of the screen (40 bytes per line). The call to the subroutine at 2100 breaks the line address up into high and low bytes and places these numbers in memory. The subroutine at 2100 computes the appropriate address four times, incrementing the address by two screen lines each time. The two calls to this subroutine (from lines 13020 and 13030) thus generate eight addresses of lines to scroll. Line 13000, besides dimensioning some strings, reads the odd looking data in lines 13045-13240 into memory. This data is called hex code, it is the hexadecimal representation (Base 16) of the machine language code. The subroutine call to line 2000 is to translate hex code into normal decimal numbers, which are then placed into memory. Although it is slow to read each hex code, translate to decimal, and place into memory (most of the long initialization at the beginning is spent doing this), it is far more efficient in terms of memory occupied by the program to use hex code instead of decimal for long listings.

The routine from 13200 to 13310 sets up the machine routine to move the diver's spear up the screen. It is (did you guess?) the one used to move "Hopper's" frog across the screen.

Finally, the lines from 13320 to 13330 set up a routine to remove the spear from the screen when it hits a target or goes off the top of the screen. This routine actually gets called twice, the first time to copy the spear image from wherever it is on the screen back to the bottom of the screen, and the second time to copy blanks to the spear position on the screen, erasing it.

The other setting up is done by the remaining subroutines. The routine at lines 10000-10112 sets up and draws the redefined characters (targets) in their rows and places the last set of numbers required by the scroll routine into the proper page six locations. These numbers (10100-10112) determine the speed and direction that the eight lines will scroll: a zero means one space left, a one means two spaces left, a two means one space right, and a three means two spaces right.

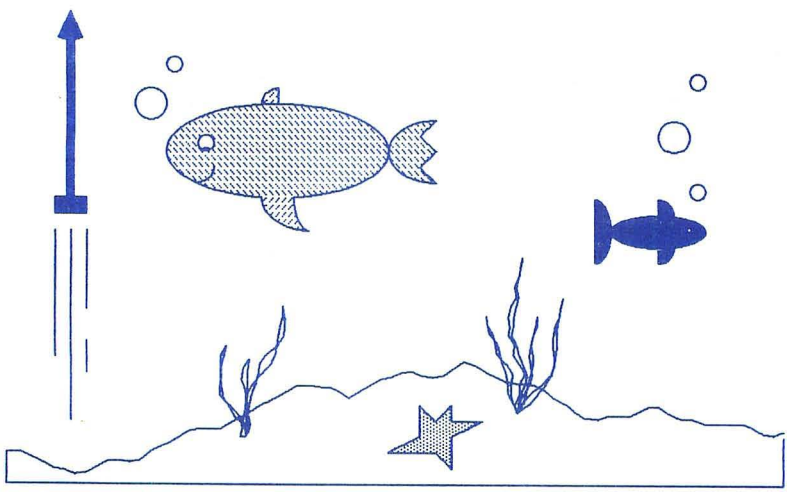
The routine at 12000-12090 sets up and draws the Player/Missile shapes and the routine starting at line 20000 sets up the redefined character set which makes up the targets. Line 20000 sets aside more or less protected memory by stepping back location 106 (RAMTOP). Normally, the computer will not place anything above the location indicated in memory location 106, so it is generally safe for storage (as with anything else, there are exceptions). Line 20010 uses a short machine language routine to download the ROM character set into

RAM where it can be modified. Finally, 20020 to 20040 POKEs the modified characters into the appropriate memory locations. The modified characters were designed using ***SoftSide's*** own character generator (October 1981, by Alan J. Zett). This fine program writes the BASIC code for inclusion into your program. It even "remembers" which characters were modified, so that code is generated only for those characters.

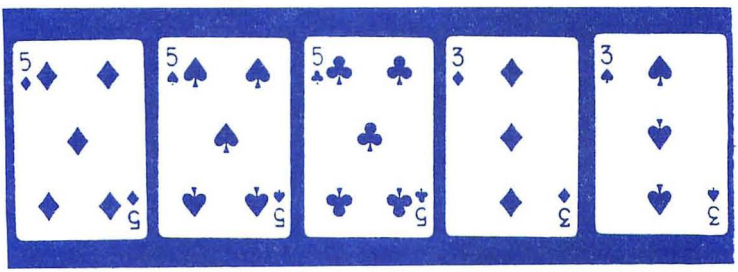
The second factor which makes a BASIC program operate smoothly is simple logic and compact code. The section of "Codfish Ball" which actually executes over and over again to give the game visuals (the "loop") is only ninteen lines of BASIC (100-260). Of these ninteen lines, only about twelve execute each time through the loop; the rest are ignored because the appropriate conditions are not met. By keeping the logic simple, the amount of code in the loop is kept small, and the program can run quite fast. BASIC simply cannot handle sophisticated (or unsophisticated, for that matter) artificial intelligence routines with any speed at all. I know. I tried to get my "Munchkin Attack" to run faster. Of course, these types of routines can be machine coded too, but the likelihood of finding ready made machine routines for things like artificial intelligence is much less than finding P/M and memory move routines.

One of the interesting points of the "loop" is line 100, which contains all those SOUND statements which generate a pitter-patter noise and slow down the program when no spear was on the screen. In the absence of a spear, all the code from lines 110 to 230 is skipped. The targets get to whizzing by too fast to play. The use of the SOUND statements slows the (spearless) program down to the point where it runs at about the same speed as when a spear is on the screen. SOUND statements were chosen over a FOR/NEXT delay loop because it gave the program more depth (no pun intended...I think). If you don't like the sound, you can change these statements, or leave them out altogether. Put the following in their place to keep the timing even:

```
FOR WAIT=1 TO 15:NEXT WAIT:GOTO 240
```



CRIBBAGE SQUARES



by Bob Whitworth and Ron O'Laughlin

Cribbage Squares is a card game for an Apple with 48K.

Cribbage Squares is a game similar to SoftSide's issue #41 *Poker Squares.* The object of the game is to create eight cribbage hands of five cards each. The eight hands will then be evaluated by the computer, and scores accumulated for each player and displayed in a scoresheet at game's end.

The computer first shuffles, turning up sixteen cards, one at a time. The player then drops the cards on a four by four grid by pressing one of the letters A-P corresponding to the small rectangles that make up the grid. As the exposed top card is played onto the grid, the computer turns up the next one.

After all sixteen cards are played on the grid, a seventeenth card will be turned up and will represent a common fifth card for each of the eight "hands" in the grid. The "hands" are the four rows across and the four columns going down. The computer will begin to count up the scores for each hand, with rows first and columns last. The total score for each player will be noted on a summary scoresheet after each game.

You can play the game as solitaire, or up to four players can compete head-to-head. In the multiple-player mode, each person plays, in turn, the same sixteen cards turned up in the same order. Of course, when one person is playing, others must not watch the cards turn up for him; this is called cheating. Each player may be allowed a different amount of time in which to play each card, providing a very effective method of handicapping.

Beware of the following hazards:

- Every time the clock runs down to zero, the player receives a five point penalty, and the clock is reset;
- If you hit an improper key (one that does not match one of the A-P squares, or one that has been previously selected) the appropriate error message appears. And remember, once a card is dropped onto the grid, you cannot pick it up or move it again.

To interrupt the game, hit the ESC key. Hitting the space bar resumes the contest. If you want to examine the scoring done by the computer after each hand, just hit the ESC key to stop the scoring and examine the hand. To speed up the scoring routines, hit the space bar after the hand has been counted, and it will speed ahead to the next hand.

Scoring for *Cribbage Squares* is identical to scoring in the game of cribbage. Every separate grouping of cards that makes a total of fifteen, a pair or a run counts separately. Three of a kind (pair royal), counts the same as three pairs, as three different sets of pairs may be formed. Four of a kind (double pair royal) counts the same as four pairs. A scoring chart follows for some of the scoring combinations of cribbage. (Figure 1.)

Strategy is very important in this game. For example, you will learn to go for combinations of fifteen in one direction, and prepare for runs and double runs in the other. As you play more and more, you will begin to get a feel for the best way to place your cards for the best possible chance for a high score.

Scoring Chart

Fifteen	Each combination of two or more cards that totals exactly fifteen scores two points. Remember that face cards count ten and all others count their numerical value.
Pair	Each different pair of cards of the same rank will score two points.
Pair Royal	Three of a kind of the same rank scores six points.
Double Pair Royal	Four of a kind of the same rank scores 12 points.
Run	Each combination of three or more cards that are in sequence scores 1 point for each card in the sequence.
Flush	Four cards of the same suit in a hand scores 4 points. If the seventeenth card turned up is of the same suit as the other cards in the hand, it also scores as a point.
His Nibs	When a jack in the hand is of the same suit as the seventeenth card, it scores one point.

Typing in the program

Enter the first listing, *Cribbage Setup.* When you run this program, it will create the hi-res picture and shape table used by *Cribbage Squares,* and save them on disk. Type in *Cribbage Squares* and save it on the same disk, so that it may access these files.

Variables

A$: Temporary string variable.
A%(*): Flags to indicate if a particular card has been dealt.
C: Total pip value of cards in hand.
C%(X): Adjusted value of card (ten max. value).
CL: Time remaining on clock.
CP%(*): Seconds per card for each player.
D%(*,*): Patterns by card for checking of fifteens.
F1%(*): Value of each card dealt.
F2%(*): Suit of each card dealt.
G4: Number of games played.
G$: Patterns by hand for checking of fifteens.
K1: Flag for type of flush in hand.
L%(*,*): Card value at board position.
M%(*,*): Suit at board position.
N: Number of players.
N$(*): Names of each player.
P: Score subtotal for each hand.
P5: Number of types of scores in the hand.
P6: Total score for the hand.
P7%(*): Accumulated score per player.
P8%(*): Total penalty points per player.
P9%(*): Total score per player.
S%(*): Number of cards in each suit for hand.
T%(*): Number of cards by card value for hand.
V: Character read from keyboard.
Y: Random number (one — 52) for card dealt.
Y$: Single character of fifteen pattern.
ZZ: Address of speaker toggle.
Z1: Row number of letter selected.
Z2: Column number of letter selected.
Z9: Timing loop variable.

```
   SS SS SS SS SS SS SS SS SS SS SS
   SS                           SS
   SS      Applesoft BASIC      SS
   SS      'Cribbage Setup'     SS
   SS   Authors: Bob Whitworth  SS
   SS        Ron O'Laughlin     SS
   SS     Copyright © 1983      SS
   SS SoftSide Publications, Inc SS
   SS                           SS
   SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on issue #47 SoftSide DV.

```
100  READ A: READ B
110  FOR I = A TO B
120  READ C
130  POKE I,C
140  NEXT
150  PRINT  CHR$ (4)"BSAVE CRIBBA
     GE SHAPES,A$6000,L607"
160  DATA 24576,25183
170  DATA 17,0,38,0,50,0,65,0,77,
     0
180  DATA 89,0,101,0,114,0,123,0,
     136,0
190  DATA 148,0,164,0,173,0,186,0
     ,199,0
200  DATA 50,1,148,1,241,1,95,2,3
     3,36
210  DATA 100,12,14,14,54,63,119,
     9,46,0
220  DATA 41,45,45,216,219,16,12,
     12,45,32
230  DATA 28,63,30,7,0,1,168,45,5
     ,32
240  DATA 28,103,12,60,63,63,0,73
     ,33,5
250  DATA 56,63,39,12,12,12,54,46
     ,0,1
260  DATA 112,45,5,32,228,63,39,4
     4,45,45
270  DATA 0,9,45,5,32,28,63,214,3
     6,100
280  DATA 12,45,5,0,33,100,12,12,
     228,58
290  DATA 63,7,0,9,45,12,228,63,2
     14,36
```

CRIBBAGE SQUARES

```
300 DATA 32,12,45,14,54,0,41,101
    ,12,60
310 DATA 63,7,32,12,45,21,46,0,3
    3,36
320 DATA 36,108,9,45,14,54,54,30
    ,63,7
330 DATA 32,36,36,0,1,112,45,5,3
    2,36
340 DATA 36,4,0,9,109,28,223,108
    ,13,36
350 DATA 228,95,191,54,7,0,33,36
    ,36,108
360 DATA 9,30,30,30,14,14,14,5,0
    ,73
370 DATA 9,45,45,45,229,59,63,12
    ,109,73
380 DATA 56,255,59,223,63,7,40,4
    5,109,109
390 DATA 45,45,5,56,63,63,63,63,
    63,63
400 DATA 63,7,40,45,45,45,45,45,
    45,45
410 DATA 45,60,63,63,63,63,63,63
    ,63,63
420 DATA 44,45,45,45,45,45,45,45
    ,45,28
430 DATA 63,63,63,63,63,63,63,39
    ,45,45
440 DATA 45,45,45,45,45,229,63,6
    3,63,63
450 DATA 63,63,103,45,45,45,45,4
    5,229,63
460 DATA 63,63,63,103,45,45,45,2
    29,63,63
470 DATA 103,45,229,39,45,0,73,9
    ,45,45
480 DATA 45,229,59,63,12,109,73,
    56,255,59
490 DATA 223,63,7,40,45,109,109,
    45,45,5
500 DATA 56,63,63,63,63,63,63,63
    ,7,40
510 DATA 45,45,45,45,45,45,45,45
    ,60,63
520 DATA 63,63,63,63,63,63,63,44
    ,45,45
530 DATA 45,45,45,45,45,45,28,63
    ,63,63
540 DATA 63,63,63,63,103,41,109,
    45,109,45
550 DATA 220,27,63,255,8,45,45,4
    5,56,63
560 DATA 63,63,44,45,45,45,28,63
    ,63,103
570 DATA 45,229,63,0,73,73,9,37,
    255,40
580 DATA 45,45,60,63,63,31,40,45
    ,45,45
590 DATA 45,60,63,63,63,63,31,20
    0,45,45
600 DATA 45,45,45,45,37,63,63,63
    ,63,63
610 DATA 63,255,40,45,45,45,45,4
    5,45,45
620 DATA 45,60,63,63,63,63,63,63
    ,63,63
630 DATA 76,45,45,45,45,45,45,37
    ,63,63
640 DATA 63,63,63,63,103,41,45,4
    5,45,45
650 DATA 60,63,63,63,63,76,45,45
    ,45,24
660 DATA 63,63,103,41,60,7,0,73,
    73,9
670 DATA 37,255,40,45,45,60,63,6
    3,31,40
680 DATA 45,45,45,45,60,63,63,63
    ,63,31
690 DATA 40,45,45,45,45,45,45,60
    ,63,63
700 DATA 63,63,63,63,31,40,45,45
    ,45,45
710 DATA 45,45,45,45,60,63,63,63
    ,63,63
720 DATA 63,63,63,44,45,45,45,45
    ,45,45
730 DATA 45,45,60,63,63,63,63,63
    ,63,63
740 DATA 63,76,45,45,109,41,45,4
    5,60,63
750 DATA 63,223,63,63,39,45,45,1
    09,41,45
760 DATA 45,220,255,219,27,103,4
    3,29,77,73
770 DATA 201,37,255,219,27,63,0,
    45,45,45
```

```
780  DATA 45,28,63,63,63,12,45,45
     ,28,63
790  DATA 12,5,0,0
800  HGR : HCOLOR= 6: HPLOT 0,0: CALL
     62454: HCOLOR= 7
810  POKE 232,0: POKE 233,96: HPLOT
     200,0 TO 200,135: SCALE= 1: ROT=
     0
820  POKE 49234,0: FOR X = 24 TO
     128 STEP 30: FOR Y = 19 TO 1
     23 STEP 31
830  HPLOT X,Y TO X + 27,Y TO X +
     27,Y + 28 TO X,Y + 28 TO X,Y
840  FOR A = 1 TO 28: HPLOT X,Y +
     A TO X + 25,Y + A: NEXT A: NEXT
     Y: NEXT X: HCOLOR= 4
850  HCOLOR= 4
860  HPLOT 20,15 TO 145,15 TO 145
     ,144 TO 20,144 TO 20,15
870  HPLOT 19,14 TO 146,14 TO 146
     ,145 TO 19,145 TO 19,14
880  HPLOT 18,13 TO 147,13 TO 147
     ,146 TO 18,146 TO 18,13
890  HCOLOR= 1
900  HPLOT 17,12 TO 148,12 TO 148
     ,147 TO 17,147 TO 17,12
910  HPLOT 16,11 TO 149,11 TO 149
     ,148 TO 16,148 TO 16,11
920  HPLOT 15,10 TO 150,10 TO 150
     ,149 TO 15,149 TO 15,10
930  HCOLOR= 3
940  HPLOT 14,9 TO 151,9 TO 151,1
     50 TO 14,150 TO 14,9
950  HPLOT 13,8 TO 152,8 TO 152,1
     51 TO 13,151 TO 13,8
960  HPLOT 12,7 TO 153,7 TO 153,1
     52 TO 12,152
970  HPLOT 12,7 TO 12,152
980  HCOLOR= 4: POKE 49235,0
990  FOR X = 0 TO 10: HPLOT X,0 TO
     X,160: NEXT X
1000  FOR Y = 0 TO 6: HPLOT 0,Y TO
     153,Y: NEXT Y
1010  FOR Y = 152 TO 160: HPLOT 0
     ,Y TO 153,Y: NEXT Y
1020  FOR X = 154 TO 279: HPLOT X
     ,0 TO X,160: NEXT X
1030  HCOLOR= 3: FOR X = 1 TO 29:
     HPLOT 189,113 + X TO 216,11
     8 + X: NEXT X
1040  HCOLOR= 1: FOR X = 1 TO 27:
     HPLOT 191,119 + X TO 214,11
     9 + X: NEXT X
1050  HCOLOR= 4
1060  HPLOT 37,31 TO 38,31: HPLOT
     36,32 TO 36,37: HPLOT 39,32 TO
     39,37: HPLOT 37,34 TO 38,34
1070  HPLOT 68,31 TO 66,31 TO 66,
     37 TO 68,37: HPLOT 67,34 TO
     68,34: HPLOT 69,32 TO 69,33:
     HPLOT 69,35 TO 69,36
1080  HPLOT 99,32: HPLOT 99,36: HPLOT
     97,31 TO 98,31: HPLOT 97,37 TO
     98,37: HPLOT 96,32 TO 96,36
1090  HPLOT 128,31 TO 126,31 TO 1
     26,37 TO 128,37: HPLOT 129,3
     2 TO 129,36
1100  HPLOT 39,62 TO 36,62 TO 36,
     68 TO 39,68: HPLOT 37,65 TO
     38,65
1110  HPLOT 69,62 TO 66,62 TO 66,
     68: HPLOT 67,65 TO 68,65
1120  HPLOT 98,63 TO 99,63 TO 99,
     67: HPLOT 97,68 TO 98,68: HPLOT
     96,63 TO 96,67: HPLOT 97,62 TO
     98,62: HPLOT 99,63
1130  HPLOT 126,62 TO 126,68: HPLOT
     129,62 TO 129,68: HPLOT 127,
     65 TO 128,65
1140  HPLOT 37,93 TO 39,93: HPLOT
     37,99 TO 39,99: HPLOT 38,94 TO
     38,98
1150  HPLOT 66,97 TO 66,98: HPLOT
     67,99 TO 68,99: HPLOT 69,93 TO
     69,96
1160  HPLOT 96,93 TO 96,99: HPLOT
     97,96: HPLOT 99,93 TO 99,94:
     HPLOT 99,98 TO 99,99: HPLOT
     98,95: HPLOT 98,97
1170  HPLOT 126,93 TO 126,99 TO 1
     25,99
1180  HPLOT 36,124 TO 36,130: HPLOT
     40,124 TO 40,130: HPLOT 37,1
     25: HPLOT 38,126: HPLOT 39,1
     25
```

CRIBBAGE SQUARES

```
1190  HPLOT 66,124 TO 66,130: HPLOT
      69,124 TO 69,130: HPLOT 67,1
      26: HPLOT 68,127
1200  HPLOT 97,124 TO 98,124: HPLOT
      97,130 TO 98,130: HPLOT 96,1
      25 TO 96,129: HPLOT 99,125 TO
      99,129
1210  HPLOT 126,124 TO 126,130: HPLOT
      127,124 TO 128,124: HPLOT 12
      7,127 TO 128,127: HPLOT 129,
      125 TO 129,126
1220  HCOLOR= 3
1230  HPLOT 172,7 TO 162,7 TO 162
      ,37 TO 172,37
1240  HPLOT 172,8 TO 163,8 TO 163
      ,36 TO 172,36
1250  HPLOT 188,7 TO 178,7 TO 178
      ,37
1260  HPLOT 188,8 TO 179,8 TO 179
      ,37
1270  HPLOT 188,7 TO 188,21 TO 17
      8,21
1280  HPLOT 187,8 TO 187,20 TO 17
      9,20
1290  HPLOT 183,21 TO 188,37
1300  HPLOT 182,21 TO 187,37
1310  HPLOT 194,7 TO 194,37
1320  HPLOT 195,7 TO 195,37
1330  HPLOT 210,7 TO 201,7 TO 201
      ,37 TO 210,37
1340  HPLOT 211,20 TO 211,8 TO 20
      2,8 TO 202,36 TO 211,36 TO 2
      11,23
1350  HPLOT 212,9 TO 212,19: HPLOT
      212,24 TO 212,35
1360  HPLOT 203,21 TO 210,21 TO 2
      10,20
1370  HPLOT 203,22 TO 210,22 TO 2
      10,23
1380  HPLOT 227,7 TO 218,7 TO 218
      ,37 TO 227,37
1390  HPLOT 228,20 TO 228,8 TO 21
      9,8 TO 219,36 TO 228,36 TO 2
      28,23
1400  HPLOT 229,9 TO 229,19: HPLOT
      229,24 TO 229,35
1410  HPLOT 220,21 TO 227,21 TO 2
      27,20: HPLOT 220,22 TO 227,2
      2 TO 227,23
1420  HPLOT 245,37 TO 245,7 TO 23
      4,7 TO 234,37
1430  HPLOT 244,37 TO 244,8 TO 23
      5,8 TO 235,37
1440  HPLOT 236,21 TO 243,21: HPLOT
      236,22 TO 243,22
1450  HPLOT 261,7 TO 251,7 TO 251
      ,37 TO 261,37 TO 261,21 TO 2
      57,21
1460  HPLOT 261,8 TO 250,8 TO 250
      ,36 TO 260,36 TO 260,22 TO 2
      57,22
1470  HPLOT 277,7 TO 267,7 TO 267
      ,37 TO 277,37
1480  HPLOT 277,8 TO 268,8 TO 268
      ,36 TO 277,36
1490  HPLOT 275,21 TO 269,21: HPLOT
      275,22 TO 269,22
1500  HPLOT 177,47 TO 167,47 TO 1
      67,62 TO 176,62 TO 176,76 TO
      167,76
1510  HPLOT 177,48 TO 166,48 TO 1
      66,61 TO 177,61 TO 177,77 TO
      167,77
1520  HPLOT 193,47 TO 183,47 TO 1
      83,77 TO 193,77 TO 193,48
1530  HPLOT 192,48 TO 184,48 TO 1
      84,76 TO 192,76 TO 192,46
1540  HPLOT 190,78 TO 192,78 TO 1
      92,79
1550  HPLOT 191,79
1560  HPLOT 209,47 TO 209,77 TO 1
      99,77 TO 199,47
1570  HPLOT 208,47 TO 208,76 TO 1
      98,76 TO 198,47
1580  HPLOT 225,77 TO 225,47 TO 2
      15,47 TO 215,77
1590  HPLOT 224,77 TO 224,48 TO 2
      14,48 TO 214,77
1600  HPLOT 215,61 TO 223,61: HPLOT
      215,62 TO 223,62
1610  HPLOT 231,77 TO 231,47 TO 2
      41,47 TO 241,62 TO 232,62 TO
      232,77
1620  HPLOT 232,61 TO 232,48 TO 2
      40,48 TO 240,61 TO 233,61
1630  HPLOT 235,62 TO 240,77: HPLOT
      236,62 TO 241,77
```

```
1640  HPLOT 257,47 TO 247,47 TO 2
      47,77 TO 257,77
1650  HPLOT 257,48 TO 248,48 TO 2
      48,76 TO 257,76
1660  HPLOT 255,61 TO 247,61: HPLOT
      255,62 TO 247,62
1670  HPLOT 273,47 TO 263,47 TO 2
      63,62 TO 272,62 TO 272,76 TO
      263,76
1680  HPLOT 273,48 TO 262,48 TO 2
      62,61 TO 273,61 TO 273,77 TO
      263,77
1690  FOR R = 1 TO 27
1700  HPLOT 168,82 + R TO 270,82 +
      R
1710  NEXT R
1720  HCOLOR= 0
1730  DRAW 14 AT 173,104
1740  DRAW 15 AT 223,104
1750  HCOLOR= 5
1760  DRAW 16 AT 198,104
1770  DRAW 17 AT 248,104
1780  HCOLOR= 6
1790  HPLOT 178,114 TO 261,114 TO
      261,151 TO 178,151 TO 178,11
      4
1800  HPLOT 176,113 TO 262,113 TO
      262,152 TO 176,152 TO 176,11
      3
1810  HPLOT 179,114 TO 179,151
1820  HPLOT 260,114 TO 260,151
1830  HPLOT 176,114 TO 176,151
1840  PRINT  CHR$ (4)"BSAVE CRIBB
      AGE BOARD,A$2000,L$1FF8"
```

LINES	STOMP CODE	LENGTH
100- 210	NB	236
220- 330	DY	374
340- 450	DY	387
460- 570	IP	394
580- 690	QO	386
700- 810	KA	366
820- 930	KJ	335
940- 1050	VW	294
1060- 1160	YP	542
1170- 1280	OW	401
1290- 1400	KN	308
1410- 1520	NL	428
1530- 1640	BC	362
1650- 1760	JF	229
1770- 1840	IS	198

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS       Applesoft BASIC       SS
SS      'Cribbage Squares'     SS
SS   Authors: Bob Whitworth    SS
SS        Ron O'Laughlin       SS
SS     Copyright © 1983        SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is available on issue #47 SoftSide DV.

Initialization.

```
100  LOMEM: 18432:D$ =  CHR$ (4)
110  PRINT  CHR$ (21);: HOME : VTAB
     12: HTAB 4: PRINT "C R I B B
      A G E   S Q U A R E S"
120  DIM N$(4),A%(52),F1%(17),F2%
     (17),L%(5,5),M%(4,4),P9%(4),
     S%(15),T%(15),CP%(4),P8%(4),
     P7%(4),D%(26,5),C%(5)
```

Read in card patterns for the checking of fifteens.

```
130  FOR X1 = 1 TO 26
140  READ G$
150  FOR X2 = 1 TO 5:Y$ =  MID$ (
     G$,X2,1):D%(X1,X2) =  VAL (Y
     $): NEXT X2
160  NEXT X1
170  DATA 11111,11110,11101,11011
     ,11100,11010,11001,11000,011
     11,01110,01101,01100
180  DATA 10111,10110,00111,00110
     ,10011,01011,00011,10101,101
     00,10010,10001,01010,01001,0
     0101
```

CRIBBAGE SQUARES

Load the game board and the shape table.

```
190  HGR : PRINT : PRINT D$"BLOAD
     CRIBBAGE SHAPES,A$4000"
200  POKE 232,0: POKE 233,64: SCALE=
     1: ROT= 0:ZZ = 49200: HOME :
     S4 = 0
210  POKE 49235,0: GOSUB 220: GOTO
     370
220  PRINT : PRINT D$"BLOAD CRIBB
     AGE BOARD"
230  RETURN
```

Shuffle the cards.

```
240  HOME : INVERSE : VTAB 23: HTAB
     26: PRINT "SHUFFLING": NORMAL
     : HTAB 25: PRINT "Press SPAC
     E": POKE 49168,0
250  IF  PEEK (49152) < 128 THEN
     Y =  RND (1): GOTO 250
```

Deal cards.

```
260  FOR X = 1 TO 17
270 Y =  INT ( RND (1) * 52) + 1:
     IF A%(Y) = 1 THEN 270
280 A%(Y) = 1:F2%(X) =  INT ((Y -
     1) / 13) + 1:F1%(X) = Y - (F
     2%(X) - 1) * 13
290  NEXT X
```

Zero out previous subtotals.

```
300  HOME
310  FOR X = 1 TO 4:P3%(X) = 0:P8
     %(X) = 0: NEXT X
320  FOR X = 1 TO 52:A%(X) = 0: NEXT
     X
330  RETURN
```

Add up patterns and print fifteen score.

```
340 C = C%(1) * D%(X5,1) + C%(2) *
     D%(X5,2) + C%(3) * D%(X5,3) +
     C%(4) * D%(X5,4) + C%(5) * D
     %(X5,5): IF C <  > 15 THEN 3
     60
350 P = P + 2: VTAB 21: HTAB 12: PRINT
     "15  -"P
360  RETURN
```

Ask for number of players.

```
370  HOME : VTAB 21: INPUT "# of
     players (1-4): ";N: IF N < 1
      THEN 390
380  IF N < 5 THEN 420
390  CALL  - 198: PRINT "Please t
     ry again"
400  FOR Z9 = 1 TO 1000: NEXT Z9
410  GOTO 370
420  FOR X = 1 TO N
```

Get players' names.

```
430  HOME : VTAB 21: PRINT "Name
     of player "X;: INPUT ": ";X$
440 X$ =  LEFT$ (X$,10):LX =  LEN
     (X$): IF LX = 0 THEN 430
450 A$ =  LEFT$ (X$,1):N$(X) =  CHR$
     ( ASC (A$) - 32 * ( ASC (A$)
     > 96)): IF LX > 1 THEN  FOR
     Y = 2 TO LX:A$ =  MID$ (X$,Y
     ,1):N$(X) = N$(X) +  CHR$ ( ASC
     (A$) + 32 * ( ASC (A$) > 64 AND
      ASC (A$) < 91 AND  PEEK (64
     435) = 6)): NEXT
460  NEXT
```

Get time limits for each player.

```
470  HOME
480  FOR X = 1 TO N
490  VTAB 20 + X: PRINT "Secs. al
     lowed/card-"N$(X)" (5-60)";:
      INPUT ": ";CP%(X): IF CP%(X
     ) > 60 THEN 510
500  IF CP%(X) > 4 THEN 540
510  CALL  - 198: VTAB 20 + X: FOR
     X1 = 1 TO 40: PRINT " ";: NEXT
     X1: VTAB 20 + X: PRINT "Plea
     se try again"
520  FOR Z9 = 1 TO 1000: NEXT Z9
530  VTAB 20 + X: FOR X1 = 1 TO 4
     0: PRINT " ";: NEXT X1: GOTO
     490
540  NEXT X
550  GOSUB 240
```

Start of play for each player.

```
560  FOR X = 1 TO N
570  FOR X4 = 1 TO 4
580  FOR X5 = 1 TO 4:L%(X4,X5) =
     0:M%(X4,X5) = 0: NEXT X5
590  NEXT X4
600  HOME : VTAB 21: PRINT "Start
      of play, "N$(X): PRINT : PRINT
     "Press SPACE": POKE 49168,0
```

```
610  IF  PEEK (49152) < 128 THEN
     610
620  POKE 49168,0: HCOLOR= 7
```

Draw square for "up" card.

```
630  FOR X6 = 1 TO 29: HPLOT 223,
     118 + X6 TO 250,118 + X6: NEXT
     X6
640  FOR X1 = 1 TO 17: HCOLOR= 4:
      DRAW F1%(X1) AT 224,126: IF
     F2%(X1) < 3 THEN 660
```

Draw "turned up" card.

```
650  HCOLOR= 5
660  DRAW (F2%(X1) + 13) AT 228,1
     43: GOTO 670
670  IF X1 = 17 THEN 920
```

Count down time, waiting for a card drop.

```
680  HOME : VTAB 22: HTAB 3: PRINT
     "Place your card.....";:CL =
     CP%(X): VTAB 22: HTAB 28: PRINT
     "Time left": HTAB 32: PRINT
     CL
690  FOR Z8 = 1 TO 65
700 V =  PEEK (49152): IF V > 128
      THEN 790
710  NEXT Z8
720 CL = CL - 1: VTAB 23: HTAB 30
     : PRINT  SPC( 6): VTAB 23: HTAB
     32: PRINT CL: IF CL <  > 3 THEN
     740
730  CALL  - 198
```

Out of time buzzer.

```
740  IF CL > 0 THEN 690
750  FOR Z1 = 1 TO 15:Z3 =  PEEK
     (ZZ) -  PEEK (ZZ) +  PEEK (Z
     Z) -  PEEK (ZZ) +  PEEK (ZZ)
     : FOR ZZ = 1 TO 5: NEXT ZZ: NEXT
     Z1
760  VTAB 22: PRINT "You have run
      out of time": PRINT "5 pena
     lty points":P8%(X) = P8%(X) -
     5
770  FOR Z9 = 1 TO 1000: NEXT Z9
780  GOTO 670
```

Check for proper card drop.

```
790  POKE 49168,0:V = V - 128 - 3
     2 * (V > 224): IF V = 27 THEN
      GOSUB 2170
800  IF V = 27 THEN 720
810  IF V < 65 THEN 830
820  IF V < 81 THEN 840
830  VTAB 22: PRINT "Please try a
     gain" SPC( 8): CALL  - 198: GOTO
     690
```

Check if square is already occupied.

```
840 V = V - 64:Z1 =  INT ((V - 1)
     / 4) + 1:Z2 = V - (Z1 - 1) *
     4: IF L%(Z1,Z2) = 0 THEN 860
850  CALL  - 198: VTAB 22: PRINT
     "Square taken--try again": GOTO
     690
```

Store card in square and remove "up" card.

```
860 L%(Z1,Z2) = F1%(X1):M%(Z1,Z2)
      = F2%(X1): HCOLOR= 7: DRAW
     F1%(X1) AT 224,126: DRAW (F2
     %(X1) + 13) AT 228,143
```

Remove selected letter from grid.

```
870  FOR X2 = 1 TO 8: HPLOT 34 +
     (Z2 - 1) * 30,X2 + 29 + (Z1 -
     1) * 31 TO 40 + (Z2 - 1) * 3
     0,X2 + 29 + (Z1 - 1) * 31: NEXT
     X2
```

Draw card in square.

```
880  HCOLOR= 4: DRAW F1%(X1) AT 2
     5 + (Z2 - 1) * 30,26 + (Z1 -
     1) * 31: IF F2%(X1) < 3 THEN
     900
890  HCOLOR= 5
900  DRAW (F2%(X1) + 13) AT 29 +
     (Z2 - 1) * 30,42 + (Z1 - 1) *
     31
910  NEXT X1
```

Scoring by row.

```
920  FOR X7 = 1 TO 4
930  HOME : VTAB 21: PRINT "Row "
     X7;":";:P5 = 0:P6 = 0
940  FOR X8 = 1 TO 5
```

```
950  IF X8 = 5 THEN 970
960 C%(X8) = L%(X7,X8): GOTO 980
970 C%(5) = F1%(17)
980  IF C%(X8) <  = 10 THEN 1000
990 C%(X8) = 10
1000  NEXT X8
1010  GOSUB 1430
1020  FOR X8 = 1 TO 4:T%(L%(X7,X8
      )) = T%(L%(X7,X8)) + 1:S%(M%
      (X7,X8)) = S%(M%(X7,X8)) + 1
      : NEXT X8
1030 T%(F1%(17)) = T%(F1%(17)) +
     1: GOSUB 1590
1040  FOR X8 = 1 TO 4
1050  IF L%(X7,X8) <  > 11 THEN 1
      080
1060  IF M%(X7,X8) <  > F2%(17) THEN
      1080
1070 P = 1: GOTO 1090
1080  NEXT X8
1090  GOSUB 1990
1100  GOSUB 2070
1110  NEXT X7
```

Scoring by column.

```
1120  FOR X7 = 1 TO 4
1130  HOME : VTAB 21: PRINT "Colu
      mn "X7;":";:P5 = 0:P6 = 0
1140  FOR X8 = 1 TO 5
1150  IF X8 = 5 THEN 1170
1160 C%(X8) = L%(X8,X7): GOTO 118
     0
1170 C%(5) = F1%(17)
1180  IF C%(X8) <  = 10 THEN 1200
1190 C%(X8) = 10
1200  NEXT X8
1210  GOSUB 1430
1220  FOR X8 = 1 TO 4:T%(L%(X8,X7
      )) = T%(L%(X8,X7)) + 1:S%(M%
      (X8,X7)) = S%(M%(X8,X7)) + 1
      : NEXT X8
1230 T%(F1%(17)) = T%(F1%(17)) +
     1: GOSUB 1590
1240  FOR X8 = 1 TO 4
1250  IF L%(X8,X7) <  > 11 THEN 1
      280
1260  IF M%(X8,X7) <  > F2%(17) THEN
      1280
1270 P = 1: GOTO 1290
1280  NEXT X8
1290  GOSUB 1990
1300  GOSUB 2070
1310  NEXT X7
```

Print penalty points.

```
1320  HOME : VTAB 22: PRINT "Pena
      lty points: "P8%(X);P9%(X) =
      P9%(X) + P8%(X): GOSUB 2100
```

Print final score.

```
1330  HOME : VTAB 22: PRINT "Fina
      l score for "N$(X)": "P9%(X)
      ;P7%(X) = P7%(X) + P9%(X): GOSUB
      2100: IF X = N THEN 1360
1340  GOSUB 220
1350  NEXT X
```

Display summary information.

```
1360  TEXT : HOME : VTAB 5: HTAB
      11: PRINT "**  S U M M A R Y
       **":G4 = G4 + 1: PRINT : HTAB
      13: PRINT "# OF GAMES: "G4
1370  VTAB 9: PRINT "Player" SPC(
      10)"Last" SPC( 5)"Accum." SPC(
      5)"Ave.": VTAB 10: PRINT " N
      ame" SPC( 11)"Game" SPC( 5)"
      Score" SPC( 5)"Score": FOR X
      1 = 1 TO 40: PRINT "=";: NEXT
      X1
1380  FOR X = 1 TO N: VTAB (12 +
      2 * X,: PRINT N$(X);: HTAB 1
      8: PRINT P9%(X);: HTAB 27: PRINT
      P7%(X);: HTAB 37: PRINT  INT
      ((P7%(X) / G4) + .5): NEXT X
```

Prompt for another game.

```
1390  VTAB 23: HTAB 8: PRINT "Pla
      y another hand? (Y/N) ";
1400  GET A$: IF A$ = "N" OR  ASC
      (A$) = 110 THEN  END
1410  IF A$ <  > "Y" AND  ASC (A$
      ) <  > 121 THEN 1400
1420 X =  FRE (0): HOME : HGR : GOSUB
      220: GOTO 470
```

Determine which card patterns to use to check for fifteens and print the fifteen score.

```
1430 X5 = 1: GOSUB 340: IF C < 15
      THEN 1570
1440  IF C > 15 THEN 1460
```

```
1450  GOTO 1560
1460  IF C%(1) + C%(2) > 15 THEN
      1480
1470  FOR X5 = 2 TO 8: GOSUB 340:
      NEXT X5
1480  IF C%(2) + C%(3) > 15 THEN
      1500
1490  FOR X5 = 9 TO 12: GOSUB 340
      : NEXT X5
1500  IF C%(3) + C%(4) > 15 THEN
      1520
1510  FOR X5 = 13 TO 16: GOSUB 34
      0: NEXT X5
1520  IF C%(4) + C%(5) > 15 THEN
      1540
1530  FOR X5 = 17 TO 19: GOSUB 34
      0: NEXT X5
1540  FOR X5 = 20 TO 26: GOSUB 34
      0: NEXT X5
1550  IF P = 0 THEN 1570
1560 P9%(X) = P9%(X) + P:P6 = P:P
      = 0:P5 = 1
1570  FOR X6 = 1 TO 15:S%(X6) = 0
     :T%(X6) = 0: NEXT X6
1580  RETURN
```

Check for runs and print the run score.

```
1590  FOR X5 = 1 TO 11
1600  IF (T%(X5) * T%(X5 + 1) * T
      %(X5 + 2)) = 0 THEN 1770
1610  IF X5 = 11 GOTO 1630
1620  IF T%(X5 + 3) > 0 THEN 1710

1630  ON (T%(X5) + T%(X5 + 1) + T
      %(X5 + 2) - 2) GOTO 1640,165
      0,1660
1640 P = 3: GOTO 1760
1650 P = 6: GOTO 1760
1660  IF T%(X5) = 3 GOTO 1700
1670  IF T%(X5 + 1) = 3 GOTO 1700
1680  IF T%(X5 + 2) = 3 GOTO 1700
1690 P = 12: GOTO 1760
1700 P = 9: GOTO 1760
1710  IF T%(X5 + 4) > 0 THEN 1750
1720  ON (T%(X5) + T%(X5 + 1) + T
      %(X5 + 2) + T%(X5 + 3) - 3) GOTO
      1730,1740
1730 P = 4: GOTO 1760
1740 P = 8: GOTO 1760
1750 P = 5
1760  VTAB 21 + P5: HTAB 12: PRINT
      "Runs -"P;P9%(X) = P9%(X) +
      P:P6 = P6 + P:P = 0:P5 = P5 +
      1: GOTO 1780
1770  NEXT X5
```

Check for pairs and print the pair score.

```
1780  FOR X5 = 1 TO 13
1790  ON T%(X5) + 1 GOTO 1840,184
      0,1800,1810,1820
1800 P = P + 2: GOTO 1830
1810 P = P + 6: GOTO 1830
1820 P = P + 12
1830  VTAB 21 + P5: HTAB 12: PRINT
      "Pairs-"P
1840  NEXT X5
1850  IF P = 0 THEN 1870
1860 P9%(X) = P9%(X) + P:P6 = P6 +
      P:P = 0:P5 = P5 + 1
```

Check for flushes and print the flush score.

```
1870 K1 = 1
1880  FOR X5 = 1 TO 4
1890  IF S%(X5) <  > 4 THEN 1930
1900 K1 = 2
1910  IF F2%(17) <  > X5 THEN 193
      0
1920 K1 = 3
1930  NEXT X5
1940  ON K1 GOTO 1980,1950,1960
1950 P = 4: GOTO 1970
1960 P = 5
1970  HTAB 12: PRINT "Flush-"P;:P
      9%(X) = P9%(X) + P:P6 = P6 +
      P:P = 0:P5 = P5 + 1
1980  RETURN
```

Check for nibs and print the nib score.

```
1990  IF F1%(17) <  > 11 THEN 201
      0
2000 P = 2: GOTO 2020
2010  IF P = 0 THEN 2060
2020  IF P5 < 4 THEN 2040
2030  VTAB 21: HTAB 24: GOTO 2050
2040  VTAB 21 + P5: HTAB 12
```

```
2050  PRINT "Nibs -"P;:P%(X) = P
      %(X) + P:P6 = P6 + P:P = 0:
      P5 = P5 + 1
2060  RETURN
```

Check for nothing and print zero score.

```
2070  IF P5 > 0 THEN 2090
2080  VTAB 21: HTAB 12: PRINT "No
      thing"
```

Print total score for hand.

```
2090  VTAB 24: HTAB 1: PRINT "Tot
      al: "P6;
```

Display score until timer ends or key is hit.

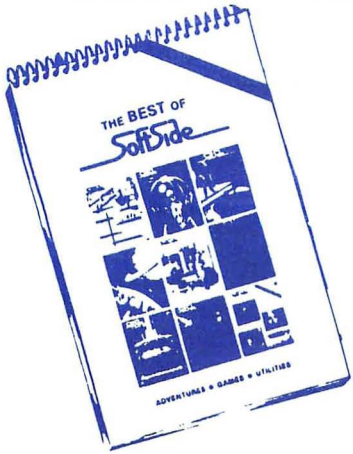
```
2100  POKE 49168,0
2110  FOR Z9 = 1 TO 100
2120 V =  PEEK (49152): IF V < 12
      8 THEN 2150
2130 V = V - 128: IF V = 27 THEN
       GOSUB 2170
2140  POKE 49168,0: GOTO 2160
2150  NEXT Z9
2160  RETURN
```

Escape key game pause.

```
2170  POKE 49168,0
2180  VTAB 23: HTAB 26: PRINT "Pr
      ess SPACE": HTAB 23: PRINT "
      to resume action";
2190  IF  PEEK (49152) < 128 THEN
      2190
2200  VTAB 23: HTAB 25: PRINT  SPC(
      13): VTAB 24: HTAB 23: PRINT
       SPC( 16)
2210  POKE 49168,0: RETURN
```

LINES	STOMP CODE	LENGTH
100- 210	IF	493
220- 330	AR	285
340- 450	EZ	421
460- 570	CH	234
580- 690	IU	341
700- 810	OW	304
820- 930	CI	444
940- 1050	QT	235
1060- 1170	XS	165
1180- 1290	TY	226
1300- 1410	NV	503
1420- 1530	QF	228
1540- 1650	SW	242
1660- 1770	ZC	240
1780- 1890	PN	188
1900- 2010	FA	169
2020- 2130	BH	211
2140- 2210	YB	144

Like SoftSide? Wait 'til you've seen our BEST!



The Best of SoftSide!

For the past four years, **SoftSide** Magazine has been bringing Apple®, Atari®, and TRS-80® owners the best in BASIC software. But now you can do even better...**The Best of SoftSide.** From all our back issues, we've selected the most useful...the most entertaining...the most fun programs **SoftSide** has ever published.

The Best of SoftSide is available in three versions...one for Apple, Atari, and TRS-80. Each contains page after page of BASIC code for adventures, simulations, practical applications, and much more.

To order your copy of **The Best of SoftSide,** fill out the coupon, and mail it along with $14.95 to **SoftSide,** 10 Northern Blvd., Northwood Executive park, Amherst, New Hampshire 03031.

The Best of SoftSide

NEW LOW PRICE!

☐ **YES...**I want to enjoy the best programs — Games, Utilities, Adventures — from SoftSide Magazine's past. Please rush my copies of The Best of SoftSide to:

Name _____

Address _____

City _____ State ____ Zip ____

Here's my order For The Best of SoftSide

Computer	Quantity Book Only $14.95 ea.	Quantity Disk Only $35.00 ea.	Quantity Book plus Disk $49.95 ea.
Apple®			
Atari®			
TRS-80® Mod I			
TRS-80® Mod III			
TOTAL COPIES	@$14.95	@$35.00	@$49.95

I'm paying by:
☐ Check # _____
☐ Money Order # _____
☐ VISA
☐ MasterCard
Total Enclosed $_____
(Foreign order please include $5 postage PAY IN USA FUNDS ONLY.)

VISA/MasterCard # _____ Exp. Date _____

Name of Cardholder _____

Signature _____

General Information

These are the standard procedures for the programs published by **SoftSide** Publications, Inc. Sometimes, a particular program does not lend itself to these procedures. Always read the specific instructions accompanying a program. They will instruct you if there are any variances from the following procedures. Also, back issues of **SoftSide** Magazine may differ in some details.

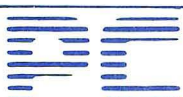
STOMP Tables

At the conclusion of each **SoftSide** listing, we include a **STOMP** Table. **STOMP** for the Atari appeared in Issue #45. **STOMP** supersedes **SWAT** as **SoftSide's** standard debugging tool to help those who type BASIC programs from the pages of **SoftSide Selections.** If you don't have **STOMP,** we'll send you a free reprint. Send a business-sized, self-addressed, stamped envelope to:

SoftSide Publications, Inc.
Department **STOMP**
10 Northern Blvd.
Northwood Executive Park
Amherst, NH 03031

Be sure to tell us what kind of computer you have.

Line Listings



IBM® PC USERS

The line listings in this booklet appear in a 40-column format. If you type LIST when your computer is displaying 40-column text (WIDTH 40), your screen should display exactly what appears in the printed listing. Be sure to use **STOMP** on your program, and get the free reprint if you don't have **STOMP.**

APPLE® USERS

The line listings in this booklet are in standard Applesoft® format, and they appear exactly as they should on your screen when you type LIST. Beginning with Issue #45, Applesoft listings in **SoftSide Selections** may have lower-case characters in them. If you have an Apple IIe or an Apple II with a lower-case adapter, you may type these listings exactly as they appear. If you have an Apple II without lower case, simply type the lower-case characters as capitals. **STOMP, SoftSide's** debugging utility, ignores the case of characters, so both ways of typing programs result in the same **STOMP** Table.

Things to watch out for when typing are:

- Lower-case characters: The Apple IIe and some Apple II's have

lower case. Many programs we publish contain lower case. Since, **STOMP, SoftSide's** debugging utility, ignores the case of characters, you may use capital letters if you so desire or if your Apple doesn't have lower case.

- REM and DATA statements: **STOMP,** like Applesoft, ignores *REM* statements. You do not have to type the text of REMs. **STOMP** also ignores the space or spaces between the keyword DATA and the first data element. Type numbers and strings in DATA statements exactly as they appear in the published listing.
- Spaces between quotes: Applesoft is a bit eccentric about how it shows these. Just list the line after you type it, and compare it to the printed listing.

Also, be sure to use **STOMP** on your program, and get the free reprint if you don't have **STOMP.**

ATARI® USERS

- The line listings in this booklet are in standard 38-column format, with special conventions for representing unprintable characters:
- You must type underlined characters, including blank spaces, in inverse video.
- When graphics or control characters are included in a string (between quotation marks), a nearby REM statement will make note of it; in such cases, graphics characters appear as the corresponding lower-case letters, and control characters appear as the corresponding unshifted key symbols. For example: The lower-case letter **s** represents a graphic cross, which you type by pressing the S key while holding down the CTRL key; the = sign represents CTRL-down-arrow, which you type by pressing and releasing the ESC key, then pressing the = key while holding down CTRL. For more information about entering control characters, refer to Appendix F and the back cover of your **Atari Reference Manual.**

There are two exceptions to our above convention: A clear-screen character (ESC SHIFT-CLEAR) appears in our listings as a right-hand brace, which looks like this: } . The other exception is that a shifted = sign appears as a broken, vertical line: ¦.

Occasionally, a program will demand that we vary from these conventions. In such a case, a nearby REM statement or the program's introductory article will clearly note the special instructions.

SoftSide®

FRONT RUNNER

CONCENTRATION

Your memory will get a serious workout with this computer version of the television game of the same name. How well can you concentrate?

AT THE CODFISH BALL

Your skill and timing are all important in this arcade-style shooting gallery.

CRIBBAGE SQUARES

Card game lovers, here is another fine game to play as solitaire or with up to three other players.

SOFTSIDE PUBLICATIONS, INC.
Amherst, New Hampshire