# GAMES IN FORTH

A good friend of mine, Mr. Li-wuu Wang, moved to the San Francisco Bay area from Los Angles. He was looking for a job. He asked me what he should do in the meantime. His major is in computer sciences. I suggested to him that he might find FORTH interesting and useful, and offered him my LSI-11 computer if he wanted to play with this language. After reading Starting FORTH and working out most of the exercises in the earlier chapters, he asked me what he should do with this language. I showed him a book, BASIC COMPUTER GAMES, edited by David H. Ahl, and asked him if he would translate or rework some of these games in FORTH. His agreed to try.

Mr. Wang spent many weekend on my computer, and literally glued his nose to the CRT screen. Most of the games collected here were his handiwork. Unfortunately, he landed a job with a local disk controller manufacturer and had to bid farewell to my computer. Otherwise, he might have just finished translating the whole book of games by now.

We can make some observations in comparing the same game written in FORTH with that in BASIC. First, the string processing capability of BASIC is quite extensive and it can handle conversation with the user very easily. Doing the same things in FORTH, one has to invoke system words like WORD and NUMBER. Second, in most of the games, integer arithmetic is sufficient and FORTH works well. If it requires anything at or above the level of square root, we have to go back and start building tools in FORTH.

Lastly, lengths of programs in FORTH are approximately the same as those in BASIC. The degree of readability of code is also the same in either language. Therefore, comparing individual programs, FORTH does not have much advantage over BASIC, as far as programming alone is concerned. However, many of the games require the same type of tools. If we build these tools once, they can be used in other programs. The real advantages in FORTH become obvious only by viewing all the games together.

In a few games, I also throw in a second version which deviates from the literal translation to take advantage of the features unique to FORTH. I hope these second versions will shed some light for the beginners and encourage them to dig deeper into the FORTH treasure box.

# ANIMAL

This game was adapted from Basic Computer Games, p. 4, Ed. D. H. Ahl, attributed originally to A. Luehrman at Dartmouth College. This program is very interesting because animals are added to the program as the game is played and it can be adapted to other areas of interests.

| | |
|---|---|
| ORIGIN | The starting block to store animal information. |
| K | A running index pointing to the line of message stored on disk. |
| KEY, EMIT | Standard terminal I/O instructions. |
| LINE | From the line number, return the address of message in disk buffer. |
| LAST | Fetch the total number of lines stored in the first cell in Block ORIGIN. |

## QUESTIONS AND ANSWERS

| | |
|---|---|
| .QA | Print the message by its line number. |
| .ANIMAL | Print all the animals in the file. |
| YES | Wait for a key stroke. Return true if 'Y' is pressed. |
| QUESTIONS | Ask questions and wait on keyboard. If 'Y' is keyed, exit the loop and store the line number in K. |
| QUERY | Wait for a text line input from the keyboard. |
| INPUT | Ask the user to type in a question that would distinguish two animals. |
| NEW-ANIMAL | Get the new animal's name and add it to the file. |
| ANSWER | Get a YES/NO answer from the user and update the line of question, in which the first two bytes point to two lines for YES/NO branching. |

## FINAL LOOP

| | |
|---|---|
| EXPAND | Move the K'th line to the end of file. |
| GUESS | Ask the user if the computer guessed the right animal. 'Y' indicates end of game. Otherwise, go the the routine to add a new animal to the file. |
| ANIMAL | The main game loop. |

```
  135 LIST

( ANIMALS, CHT, 23-NOV-82)
140 CONSTANT ORIGIN
VARIABLE K
: KEY     0 'S 1 EXPECT ;
: EMIT    'S 1 TYPE DROP ;
: LINE    ( N --- A )    16 /MOD   ORIGIN + BLOCK
          SWAP 64 * +    ;
: LAST    ( --- N )      ORIGIN BLOCK @ ;
: .QA     ( N --- )      LINE 2+ 62 -TRAILING TYPE ;
: .ANIMALS ( --- )       CR ." ANIMALS I ALREADY KNOW ARE:"
          CR   0   LAST 0 DO
          I LINE @   0= IF   I .QA 3 SPACES   1+ THEN
          DUP 5 =    IF CR DROP 0 THEN   LOOP    DROP   ;
: YES    ( --- F )       KEY 89 = ;



  136 LIST

( ANIMALS, II, CHT, 23-NOV-82)
: QUESTIONS ( --- )
          1   BEGIN   DUP LINE @   IF CR
          DUP .QA   LINE   ." ?"   YES + C@   AGAIN   K !   ;
: QUERY   S0 @ 64 EXPECT   0 >IN !   ;
: INPUT   CR ." PLEASE TYPE IN A QUESTION THAT WOULD DISTINGUISH
 A "  CR   HERE COUNT TYPE   ."  FROM A "   K @ .QA   ."  :"
          QUERY ;
: NEW-ANIMAL   CR ." THE ANIMAL YOU WERE THINKING WAS A :"
          QUERY 1 WORD 1+   LAST 1+ LINE   0 OVER !
          2+ HERE C@  MOVE   UPDATE   ;
: ANSWER   CR ." FOR A "   HERE COUNT TYPE
          ."  THE ANSWER WOULD BE :"
          1 WORD 1+   K @ LINE 2+   HERE C@ MOVE   UPDATE
          LAST DUP 1+ YES 0= IF SWAP THEN   256 * +   K @ LINE
          !   2 0 LINE +!      UPDATE   ;


  137 LIST

( ANIMALS, III, CHT, 23-NOV-82)
: EXPAND   K @ LINE   LAST LINE   64 MOVE   ;
: GUESS    CR ." IS IT A "   K @ .QA   ." ?"
           YES CR   ABORT" --- TYPE 'ANIMAL' TO TRY AGAIN."
           NEW-ANIMAL   EXPAND   INPUT   ANSWER   ;
: ANIMAL   BEGIN   CR CR
           ." ARE YOU THINKING OF AN ANIMAL?"
           YES IF   QUESTIONS   GUESS
           ELSE   .ANIMALS   ( FLUSH )   THEN
           CR CR   0 END   ;
EXIT
```

PRINT THE FILE

PRINT  Print the contents of the entire file. The first two bytes are pointers to other lines, and they are printed separately. Because of these two bytes, the file cannot be listed using the regular LIST command.

SOME COMMENTS

When I first took on the task of translating BASIC games into FORTH, I was very optimistic in that I could write the same game in much shorter codes, and really make the FORTH programs outshine their BASIC brothers. After a number of tries, I was not so sure of myself. The string commands in BASIC are very efficient in doing user interfacing. In comparison, I have to use QUERY, WORD, NUMBER, etc, to do the same thing.

The FORTH programs are not much better than the BASIC programs in length or in clarity. I am sure the FORTH programs will run much faster, but speed is not a very important concern in these games. I had to be satisfied in that I had done some examples to compare these two languages, without proving that FORTH is superior than BASIC.

```
 138 LIST

( ANIMALS, IV, CHT, 23-NOV-82)
: PRINT   LAST 0 DO   CR   I LINE DUP  C@ 3 U.R   1+ C@ 3 U.R
          I .QA   LOOP   ;
EXIT
```

```
 139 LIST
```

```
 140 LIST

DOES IT SWIM
DOES IT HAVE SCALES
DOES IT LIKE PEANUTS

IS IT STREAMLINED
FISH
```

## CALENDAR

This program is adopted from BASIC COMPUTER GAMES, p. 37, attributed to Geoffrey Chase of Abbey, Portsmouth, RI. To print a year's calendar, you must specify the day of 1 Jan and the year.

MONTH>DAY Given the month ( 0-11), it returns the days in that month by looking up the table MONTH>DAYS above.

MONTHS A super string containing the names of the months.

STARS Print out a string of '*'.

TITLE Print the header of a month specified on stack.

SKIP-DATE Skip N columns in printing the first day of month.

WEEKS Print days of a month in the 7 column format.

NEXT-MONTH From day-of-year and month, calculate the day of 1st-of-month for next month.

MONTHS Given 1st-day-of-year, print out a 12 month calendar.

HEAD Print the header for the year calendar.
?LEAP Return true if processing a leap year.
PLUS-ONE Assign 29 days to Febuary in the leap year.
CALENDAR The final word doing all the processings to print out a year calendar. The day of 1 January and the year in AD must be given on the data stack.

```
   96 LIST

( CALENDER )
CREATE  MONTH>DAYS  31 , 28 , 31 , 30 , 31 , 30 ,
                    31 , 31 , 30 , 31 , 30 , 31 ,

: MONTH>DAY    ( MONTH ---)    2 * MONTH>DAYS + @ ;

: MONTHS    ." JANUARY  FEBRUARY  MARCH    APRIL     MAY       JU
NE     JULY    AUGUST SEPTEMBER OCTOBER  NOVERBER DECEMBER" ;

: .MONTH    9 *  [']  MONTHS + 3 +  9  TYPE ;

: STARS    ( N --- )    0 DO 42 EMIT LOOP ;
```

```
   97 LIST

( CALENDER )
: TITLE    ( MONTH --- ) 3 CRS  27 STARS  .MONTH  36 STARS CR CR
   8 SPACES ." S"  8 SPACES ." M"  8 SPACES ." T"  8 SPACES
   ." W"  8 SPACES ." T"  8 SPACES ." F"  8 SPACES ." S" CR CR
   72 STARS  CR CR ;

: SKIP-DATE    ( N --- )    DUP ?DUP  IF  9 * SPACES THEN ;

: WEEKS    ( D-Y D/M --- D-Y)  1+ 1 DO  I 9 U.R  DUP I + 7 MOD
                0= IF  CR CR  THEN  LOOP ;

: NEXT-MONTH    ( MONTH --- DAY)    MONTH>DAY  +  7 MOD ;
```

```
   98 LIST

( CALENDAR     con't )
: MONTHS     ( 1ST-DAY-OF-YEAR ---, 0 FOR SUNDAY)
            12 0 DO I  TITLE  SKIP-DATE
            I MONTH>DAY WEEKS  I  NEXT-MONTH  LOOP DROP ;

: HEAD    ( Y ---)  20 CRS  25 SPACES  .  ." CALENDAR "  3 CRS ;
: ?LEAP   ( Y --- F)    4 MOD  0= ;
: PLUS-ONE    ( F ---)    IF 29 ELSE 28 THEN  2 MONTH>DAYS + ! ;
: CALENDAR    ( 1ST-DAY YEAR --- )
            DUP HEAD  ?LEAP  PLUS-ONE  MONTHS CR ;

6 1983 CALENDAR
```

## CALENDAR, 2ND VERSION

The BASIC CALENDAR works fine. However, you have to look up which day is the Jan. 1 and tell the computer before it does the printing. This version can determine that day automatically.

| | |
|---|---|
| JULIAN | The Julian day of the first day in the year. This Julian calendar starts at Jan. 1, 1949. |
| 4YEARS | Total days in four years, including a leap year. |
| DAY | An array of the first day of each month, offset from Jan. 1. |
| DAY0 | A copy of DAY, modified for leap year if the year to be processed is a leap year. |
| YEAR | Calculate the Julian day of Jan. 1 and prepare DAY0. |
| EMIT | Standard EMIT. |
| STARS | Print a string of stars. |

### PRINT ONE MONTH

| | |
|---|---|
| HEADER | Print the header for the month identified on stack. |
| BLANKS | Skip columns according to the day of the first day in this month. |
| .DAYS | Print the days in a month in the column format. |
| MONTH | Print one month calendar. |

### PRINT CALENDAR

Give the year to CALENDAR and it will print the year's calendar, taking care of leap years and days all by itself.

```
( NEW CALENDAR, 23-NOV-1981, CHT)
VARIABLE JULIAN  ( JULIAN DAY OF 1ST DAY IN A YEAR)
1461 CONSTANT 4YEARS ( DAYS IN 4 YEARS)
VARIABLE DAY   0 DAY !
   31 , 59 , 90 , 120 , 151 , 181 , 212 , 243 , 273 , 304 ,
   334 , 365 ,
VARIABLE DAY0   24 ALLOT
: YEAR   ( year --- )
         DAY DAY0 26 MOVE
         1949 - 4YEARS 4 */MOD   365 - JULIAN !
         3 = IF   DAY0 4 + DUP 22 + SWAP
         DO   1 I +!  2 +LOOP    THEN   ;
: EMIT   'S 1 TYPE DROP   ;
: STARS   0 DO 42 EMIT LOOP   ;
```

```
( PRINT CALENDAR, 24-NOV-81, CHT)
: HEADER   ( N --- )   CR CR 1- 8 *   199 BLOCK DUP
          896 + ROT +   28 STARS   8 TYPE 28 STARS CR CR
          832 + 64 TYPE CR CR   64 STARS CR CR   ;
: BLANKS   ( N --- )   1- 2* DAY0 + @   JULIAN @ +
          7 MOD 8 * SPACES   ;
: .DAYS   ( N --- )   1- 2* DAY0 + 2@   SWAP
          OVER - 0 DO   I OVER +
          JULIAN @ +   7 MOD   0= IF CR CR THEN
          I 1+ 8 U.R   LOOP   CR CR DROP   ;
: MONTH   ( N --- )   DUP HEADER DUP BLANKS DUP .DAYS   ;
EXIT

     SUN      MON      TUE      WED      THU      FRI      SAT
JANUARY FEBUARY   MARCH   APRIL   MAY      JUNE    JULY    AUGUST
SEPTEMBROCTOBER NOVENBERDECEMBER
```

```
( PRINT CALENDAR, 24-NOV-81, CHT)
: CALENDAR ( YEAR --- )
          YEAR   13 1 DO I MONTH LOOP ;

1981 CALENDAR
```

## DEPTH CHARGE

In this program you are the captain of USS Computer. Your mission is to destroy an enemy submarine. You will select the size of the 'cube' of water you wish to search. You have 7 depth charges to use.

GREET — Clear screen and print greetings.

DIMENSION — Ask the user to specify the size of the cube of water to search. This number is used to place the submarine and three coordinates are returned.

RINGS — Ring the bell on the terminal.

BACKS — Backspace the cursor so that entered numbers are overlaid on the output message.

AIM — Ask the user to input three numbers in the form of xx,yy,zz to specify the target point for a depth charge.

SHOOT — Compare the target point with the submarine coordinates. Return false if they match.

?BOOM — Reverse the sense of SHOOT.

CHEERS — Report a hit and clear the stack.

?NO-MORE — Return a false flag if the user answer the query with a 'Y'.

DEPTH-CHARGE — The game's main loop.

```
  102 LIST

( DEPTH CHARGE )
: GREET   20 CRS  ." DEPTH CHARGE"  5 CRS  ;

: DIMENSION   ( --- x y z )
      ." DIMENSION OF SEARCH AREA? "
      S0 @ 5 EXPECT  0 >IN ! 0 BLK !  1 WORD NUMBER
      DUP DUP CHOOSE  ROT CHOOSE  ROT CHOOSE
      CR CR ." GOOD LUCK ! " ;

: RINGS   ( N --- )  0 DO 10000 0 DO LOOP 7 EMIT LOOP ;
```

```
  103 LIST

( DEPTH CHARGE   con't )
: BACKS    ( n --- )   0 DO  8 EMIT LOOP ;
: AIM   ( n --- ns ew dp )
      2 CRS ." TRIAL # "  . ." : NS,EW,DP"  8 BACKS
      DUP 2OVER ROT  S0 @ 15 EXPECT  0 >IN !  44 WORD NUMBER
      44 WORD NUMBER  1 WORD NUMBER  CR ;

: SHOOT   ( x y z ns ew dp --- x y z f )
   ." SHOOT REPORT : " SWAP  2SWAP
  - DUP >R ?DUP IF 0< IF ." north" ELSE ." south" THEN THEN
  ROT - DUP >R ?DUP IF 0< IF ." west" ELSE ." east" THEN THEN
  - DUP ?DUP IF ."  AND too " 0< IF ." high." ELSE ." low."
  THEN THEN  R> R>  OR OR  ;
```

```
  104 LIST

( DEPTH CHARGE   con't )
: ?BOOM   0= ;
: CHEERS   2 CRS  ." B O O M ! ! "  RINGS
          ."  YOU GOT IT IN " . ." TRIALS ! "  2DROP DROP ;
: ?NO-MORE    5 CRS  ." ANOTHER GAME ? "  KEY  89 -  2 CRS ;

: DEPTH-CHARGE    GREET  BEGIN  DIMENSION
                  7 1 DO  I AIM  SHOOT
                  ?BOOM IF I  I' I - CHEERS LEAVE THEN LOOP
                  ?NO-MORE  END  ." BYE !" ;

DEPTH-CHARGE
```

## GAME OF LIFE

This game was adapted from that published in InforWorld, Vol. 4, p. 33, 11-Oct-1982, by M. Swaine. Only minor changes were made to display the life map on the ADDS Viewpoint CRT.

| | |
|---|---|
| TOP | Clear the CRT screen and home the cursor. |
| LGL | Return a true flag if the location is legal or in range. |
| RCA | From map coordinates return the map address. |
| ON | Initiate a life point on map. |
| OF | Kill a life point. |
| N | A macro to set DO range. |
| CLEAR | Clear the map to zero. |
| DISPLAY | Print the map contents on CRT. The map is allocated above PAD. |

| | |
|---|---|
| NBR | From a location calculate its neighboring life point. |
| BIRTH | Add the location to the stack if this location has two neighbors. |
| DEATH | Add this location to the stack if this location has more than 3 or less than 2 neighbors. |
| FLIP | Flip the state of a location. |
| MARK | Go through the entire map and pile all the locations on the stack to be flipped. |
| GENERATION | Renew the life map n times and display them on CRT |

```
  120 LIST

( GAME OF LIFE, M. SWAINE, INFOWORLD 4, 11-OCT-82, P. 33)
24 CONSTANT HT   80 CONSTANT WD              HEX
: TOP   1B EMIT 59 EMIT 20 EMIT 20 EMIT ;    DECIMAL
: LGL   ( LOC --- F )    PAD - DUP 0 > SWAP 1920 < AND ;
: RCA   ( X Y --- LOC)   PAD +   SWAP 80 * + ;
: ON    ( X Y --- )      RCA DUP LGL IF 1 SWAP C!
                         ELSE DROP THEN ;
: OF    ( X Y --- )      RCA DUP LGL IF 0 SWAP C!
                         ELSE DROP THEN ;
: N   OVER + SWAP ;
: CLEAR   PAD 1920 ERASE ;
: DISPLAY TOP PAD 1839 N
      DO I C@ IF 64 EMIT ELSE SPACE THEN LOOP TOP ;



  121 LIST

( GAME OF LIFE, CONTINUED)
: NBR ( LOC -- N )   -81 N -80 N -79 N -1 N 1 N 79 N 80 N 81 N
       DROP 0 8 0 DO   SWAP DUP LGL   SWAP C@  AND
       IF 1+ THEN   LOOP ;
: BIRTH ( N LOC --- LOC N+1 OR N )
       DUP NBR 2 =   IF SWAP 1+ ELSE DROP THEN ;
: DEATH ( N LOC --- LOC N+1 OR N )
      DUP NBR DUP 3 > SWAP 2 < OR IF SWAP 1+ ELSE DROP THEN ;
: FLIP  ( LOC --- ) DUP C@  IF 0 SWAP C! ELSE
       1 SWAP C! THEN ;
: MARK  ( --- LOC'S N ) 0   PAD 1920 N DO I
       DUP C@  IF DEATH ELSE BIRTH THEN   LOOP ;
: GENERATION ( N --- )  0 DO MARK 0 DO FLIP LOOP
       DISPLAY TOP I . LOOP ;



  122 LIST
```

## GAME OF LIFE, REVISED

The GAME OF LIFE by M. Swaine ran well. However, it took about 20 seconds to update one generation and I was not very patient to wait that long. Though there are lots of ways to cut corners in this program, the speed of execution was not significantly improved. A radical approach was taken here, using a second map to hold update information. This cuts down the run time to about 4 seconds, which is tolerable.

CLEAR-EDGES The edges on the map are cleared of any life. This strategy allows the map to be added algebraically with lateral and vertical shifts, generating the updating information in one pass.

NEIGHBORS Shift and add the life map in PAD to the status map in PAD+1920, which will then contain the numbers of neighbors for each corresponding location in life map.

LIFE? Compare locations in life map and status map and determine the life points for the next generation.

GENERATION Renew the life map n times.

```
  126 LIST

( GAME OF LIFE, REVISED, 30-AUG-83, CHT)

: CLEAR-EDGES   PAD 80 ERASE   PAD 1840 + 80 ERASE
      PAD 80 +   22 0 DO   0 OVER C!   0 OVER 79 + C!
      80 +   LOOP   DROP   ;

: NEIGHBORS   PAD 80 +   22 0 DO   80 0 DO
      >R   I 1+ C@   I 1 - C@ +   I 80 - C@ +   I 80 + C@ +
      I 79 + C@ +   I 81 + C@ +   I 79 - C@ +   I 81 - C@ +
      I 1920 + C!   R> 1+   LOOP LOOP   DROP   ;
```

```
  127 LIST

( GAME OF LIFE, REVISED, 30-AUG-83, CHT)

: LIFE?   PAD 80 +   22 0 DO   80 0 DO
      >R   I 1920 + C@   I C@
      IF   2 > IF   0 I C!   THEN
      ELSE 2 = IF   1 I C!   THEN   THEN
      R> 1+   LOOP   LOOP   DROP   ;

: GENERATION ( N --- ) 0 DO NEIGHBORS LIFE? CLEAR-EDGES
      DISPLAY LOOP ;
```

```
  128 LIST
```

## GUESS A NUMBER

This game was in D. H. Ahl's BASIC COMPUTER GAMES, p. 75, attributed to Walt Koetke of Lexington, Mass.

The computer first asks you to choose a limit and uses a random number routine to get a number between 0 and this limit. You must then try to guess the number.

The random number routine is from Leo Brodie's STARTING FORTH p. 265.

GREET — Clear the screen and print messages.

GET-NUMBER — Ask the user to input an integer number and push it on the data stack. S0 @ gives the address of the terminal input buffer.

GET-LIMIT — Get the number limit with proper query messages.

CHECK — Check the guessed number with that produced by the computer. Return a flag for looping and the target number for next guess.

?NO-MORE — Returns a true flag if use type in "Y" in response to the printed message.

GUESS — Main loop of this game, actually a doubly nested loop.

```
   90 LIST

( RANDOM NUMBER GENERATOR )
VARIABLE REN        HERE REN !
: RANDOM    REN @ 31421 *  6927 +  DUP REN ! ;
: CHOOSE    RANDOM U* SWAP DROP ;


: CRS    0 DO CR LOOP ;

: -TEXT    ( SOURCE COUNT DESTINATION --- FLAG)
      >R >R >R    0    R> R> R>    DUP ROT + SWAP DO
      DUP C@ I C@ -    ?DUP IF ROT DROP SWAP LEAVE THEN
      1+    LOOP DROP    ;
```

```
   91 LIST

( GUESS )
: GREET    30 CRS  18 SPACES  ." GUESS "    CR
   ." THIS IS A NUMBER GUESSING GAME. I'LL THINK "  CR
   ." OF A NUMBER BETWEEN 0 AND ANY LIMIT YOU     "  CR
   ." WANT ( IT SHOULD SMALL THAN 65000 ). THEN   "  CR
   ." YOU HAVE TO GUESS WHAT IT IS. "  CR CR    ;
: GET-NUMBER  S0 @ 10 EXPECT    0 >IN ! 0 BLK ! 1 WORD NUMBER ;
: GET-LIMIT    ( --- N )    ." WHAT LIMIT DO YOU WANT? "
   GET-NUMBER  DUP  CHOOSE  SWAP CR CR
   ." I'M THINK A NUMBER BETWEEN 0 AND " . CR
   ." NOW YOU TRY TO GUESS WHAT IT IS. " CR   ;
: CHECK    ( N GUESS --- N TRUE IF MISS, FALSE IF HIT)
          OVER -   DUP 0 > IF ." TOO HIGH! " ELSE
                   DUP 0< IF ." TOO LOW!   " ELSE
                   ." CORRECT!! " SWAP DROP THEN THEN  CR ;
: ?MATCH    ( N --- FLAG)    0= ;
```

```
   92 LIST

( GUESS     con't )
: ?NO-MORE    CR CR   ." DO YOU WANT PLAY AGAIN? "
              KEY 89 -   CR CR ;
: GUESS     GREET  BEGIN  GET-LIMIT
                      BEGIN  GET-NUMBER  CHECK
                      ?MATCH  END
                   ?NO-MORE END
            ." BYE-BYE "    ;
GUESS
```

## GUNNER

GUNNER asks you to specify the degrees of elevation of your weapon. 45 degrees provides maximun range. You get up to five shots to destroy the enemy before he destroys you. This game was translated from BASIC COMPUTER GAMES, p. 77, originated by Tom Kloos, Portland, Or., and modified by D. Ahl.

CHOOSE ( n --- n1) Generate a random number between 0 and n. According to L. Brodie, Starting FORTH, p. 265.

SIN ( n --- sin) Given an angle in radian*10000, return its sine value also multiplied by 10000. According to J. Bumgarner, FORTH Dimensions, Vol. IV, p. 7, (1982).

GREET Print greetings.
RANGE Maximun range.
TARGET Target distance within the maximun range.
#SHOT Rounds already fired.
SET-RANGE Select maximum range between 0 and 20000 yards.
.TARGET Select the target distance.
ANGLE Integer part of the input elevation angle.
FRACT Fractional part of the angle.
COUNT Digits after the decimal point in the fraction.
READ Input the angle and fill ANGLE, FRACT, and COUNT.
SHOOT Read input elevation and increment #SHOT.
BEEP Ring the bell.

** ( n1 n2 --- n3 ) Exponentiation.

SCALE ( --- angle) Calculate angle in radian*10000 using data stored in ANGLE, FRACT, and COUNT.
BOOMED ( --- distance) Compute the distance of the round.
?POWER ( n ---) If n=5, your are dead.

?HIT Access the result of firing and print out appropriate messages.

.ROUNDS Message output.

```
( LOAD    REN  CRS  SIN )

VARIABLE  REN    HERE  REN !
: RANDOM  REN @  31421 *   6927 +  DUP  REN ! ;
: CHOOSE  RANDOM U* SWAP DROP ;

: CRS      ?DUP  IF 0 DO CR LOOP THEN ;

10000  CONSTANT 10K      VARIABLE   XS
: L       XS @  SWAP /  MINUS  10K  */ 10K  +  ;
: REDUCE  DUP 15707 >  IF  31414 SWAP -  THEN ;
: SIN     REDUCE  DUP DUP 10K */  XS ! 10K 72 L  42 L  20 L  6 L
          10K  */  ;

61 LOAD  62 LOAD  63 LOAD
```

```
( GUNNER )
: GREET   PAGE  23 SPACES    ." GUNNER "  3 CRS
      CR  ." YOU ARE THE OFFICER-IN-CHARGE, GIVEN ORDERS TO A "
      CR  ." GUN CREW, TELLING THEM THE DEGREES OF ELEVALUE "
      CR  ." YOU ESTIMATE WILL PLACE A PROJECTILE ON TARGET "
      CR  ." A HIT WITHIN 100 YARDS WILL DESTROY THE TARGET." ;
VARIABLE   RANGE        VARIABLE   TARGET       VARIABLE  #SHOT
: SET-RENGE   200  100 CHOOSE *  RANGE !  0 #SHOT !    3 CRS
    ." MAX RANGE OF YOUR GUN IS " RANGE ? ." YARDS."  3 CRS ;
: .TARGET    RANGE @ CHOOSE   TARGET !
    ."       DISTASNT TO THE TARGET IS " TARGET ? ." YARDS." ;
VARIABLE   ANGLE        VARIABLE   FRACT        VARIABLE  COUNT
: READ   S0 @ 10 EXPECT 0 BLK ! 0 >IN !  46 WORD  NUMBER ANGLE !
         1 WORD  DUP C@  COUNT !  NUMBER  FRACT ! ;
: SHOOT  2 CRS  ." ELEVATION ? "  READ   1 #SHOT +! CR  ;
: BEEP   3 0 DO  7 EMIT  10000 0 DO LOOP  LOOP ;
```

```
( GUNNER    Con't )
: **   ?DUP IF  1- ?DUP IF  OVER ROT ROT  0 DO OVER * LOOP
       SWAP DROP THEN  ELSE  DROP 1 THEN  ;
: SCALE  ANGLE @ 17453 100 */
         FRACT @ ?DUP IF 17453 100 */  10 COUNT @ ** / + THEN ;
: BOOMED   SCALE 2 *  SIN  RANGE @ 10K */ ;
: ?POWER   5 =  IF 2 CRS ." BOOM !!!!  YOU'VE JUST BEEN DESTROY"
           BEEP  0 #SHOT ! THEN  ;
: ?HIT     0  ANGLE @ DUP  89 >
    IF     ." MAX ELEVALUE IS 89 DEGREE."   DROP
    ELSE   1 < IF ." MIN ELEVELUE IS 1 DEGREE."
    ELSE   BOOMED TARGET @ -  DUP  ABS  101 <   IF  DROP 1+
           ." **** TARGET DESTROY **** "
    ELSE   DUP  0 >  IF ." OVER "  ELSE ." SHORT " THEN  ABS
           ." OF TARGET " . ." YARDS. "  THEN THEN THEN  ;
: .ROUNDS   . ." ROUNDS OF AMMUNITION EXPENDED " ;
```

?BE-HIT ( n --- ) If n<4 and still alive, print warning.

SCORE Examine total rounds fired and make comments.

?NO-MORE ( --- f ) End of game?

GUNNER Main game loop. Four targets and 5 rounds for each.

```
( GUNNER     Con't )
: ?BE-HIT    4 = #SHOT @ NOT OR DUP IF ELSE 3 CRS ." ATTENTION!"
          ."  MORE EMENY ACTIVITIES ..... "  THEN  CR  ;
: SCORE  3 CRS #SHOT @ ?DUP IF ." TOTLE ROUNDS EXPENDED WERE: "
         DUP . CR  18 <  IF ." NICE SHOOTING ! ! "  ELSE
         ." BETTER GO BACK FOR REFRESH TRAINING "  THEN THEN ;
: ?NO-MORE  4 CRS  ." TRY AGAIN (Y/N)? "  KEY 89 -  CR DUP 0=
       IF  ." OK, RETURN TO BASE." ELSE ." BYE!" THEN 3 CRS ;
: GUNNER   GREET  BEGIN  SET-RANGE
             5 0  DO  .TARGET
                  6 0 DO  SHOOT  ?HIT IF I 1+ .ROUNES
                  LEAVE ELSE  I  ?POWER  THEN  LOOP
             I ?BE-HIT IF LEAVE THEN  LOOP
           SCORE  ?NO-MORE  END ;
```

## HELLO

This is a conversational program in which the computer will dispense advises on various problems such as sex, health, money, and job. It is adopted from the same named program in BASIC COMPUTER GAMES, p. 82, attributed to David Ahl.

| | |
|---|---|
| NAME | A string variable storing the name of the user. |
| ANSWERS | A super string. The substrings therein are used to be compared with the user inputs to pick up the desired responses. |
| NAMED | Get the user's name from terminal and store it in NAME. |
| .NAME | Print the name stored in NAME. |

| | |
|---|---|
| ANSWER | ( --- here count) Accept a string from the terminal and leave the address and byte count on the stack. |
| CHECK | ( addr count --- n ) Compare the string at addr with the substrings in ANSWERS and return the string number. If no match, return 6. |
| WRONG | A message output. |
| ?RIGHT | ( n --- ) A yes-no-else response. 4 for no, 5 for yes, and other number causes WRONG response. |
| MORE | A message output. |
| QUESTION | A message output. |

| | |
|---|---|
| GREET | Print out greeting messages, ask the user's name and store it in NAME . It also asks a yes/no question and waits for user's response. |

```
  108 LIST

( HELLO , NAMES )
VARIABLE  NAME 38 ALLOT

: ANWSERS    ."  SEX    JOB    MONEY HEALTHNO     YES    XXXXXX" ;

: NAMED    S0 @ 40 EXPECT  0 >IN !  0 BLK !
           1 WORD COUNT  NAME 40 BLANK  NAME SWAP MOVE ;

: .NAME     NAME 40 -TRAILING TYPE ;
```

```
  109 LIST

( HELLO , ANSWER & CHECK )
: ANWSER   S0 @ 10 EXPECT  0 >IN ! 0 BLK !  1 WORD COUNT   CR ;
: CHECK   7 0 DO 2DUP ['] ANWSERS 4 + I 6 * + SWAP ROT -TEXT 0=
      IF I LEAVE ELSE I 6 = IF I THEN THEN LOOP ROT ROT 2DROP ;
: WRONG  ." JUST A SIMPLE 'YES' OR 'NO' PLEASE " .NAME ." ? ";
: ?RIGHT DUP 4 = ?DUP IF ." Oh, I'm sorry to hear that, " .NAME
     ." , May be we can" CR ." brighten up your visit a bit."
     SWAP DROP ELSE 5 = DUP IF ." I'm glad to hear that. "
     ELSE WRONG THEN THEN  ;
: MORE ." Any more problems want to solve, " .NAME ." ?" ;
: QUESTION   ." What kind ( SEX, JOB, MONEY, HEALTH )? " ;
```

```
  110 LIST

( HELLO   , GREET )
: GREET   20 CRS  25 SPACES  ." HELLO "  5 CRS
     ." Hello! My name is creating computer. "  3 CRS
     ." What's your name?  " NAMED CR
     ." Hi there! " .NAME ." , Are you enjoying youself here? "
     BEGIN  ANWSER CHECK  ?RIGHT  END  ;
```

```
.HEALTH    Advises on health.
.JOB       Advises on job.
.MONEY     Advises on money.
```

```
.MUCH      One advise on sex.
.LITTLE    The other advise on sex
.SEX       Query and decision in the problems of sex.
```

```
PROBLEMS   Continuation of the initial greeting.
REPLY      ( n --- )   The case type decision process to go
           for the correct response.
?MORE      ( --- f )   A 'YES' response on the terminal causes
           the program to loop back.  Other responses will
           terminate the loop.
HELLO      The main game loop.
```

```
  111 LIST

( HELLO   HEALTH, JOB, MONEY )
: .HEALTH   ." My advise to you, " .NAME ." , is: "  CR
     ."     1. Take two tablets of aspirin "  CR
     ."     2. Drink plenty of fluids " CR
     ."     3. Go to bed ( along ) "    CR CR  ;

: .NAM   .NAME ." .  "   ;
: .JOB   ." I can sympathize with you " .NAM ." I have to " CR
     ." work very long every day with no pay. My advice to " CR
     ." you. is to open a rental computer store. " CR CR ;

: .MONEY  ." Sorry! " .NAM ." I'm broke too. Why don't you " CR
   ." sell enclopadias or marry someone rich or stop eating " CR
   ." So you won't need so much money? " CR CR ;



  112 LIST

( HELLO   MUCH, LITTLE, SEX )
: .MUCH  ." You call that a problem?!! I should have such " CR
   ." problems. If it bothers you, " .NAME ." , take a cold " CR
   ." shower. " ;
: .LITTLE  ." Why are you here! " .NAME ." , You should be " CR
   ." in Tokyo or New York or Amsterdam or some place " CR
   ." with some action. "  ;
: .SEX    ." Is your problem TOO MUCH to TOO LITTLE ( M/L )? "
   BEGIN KEY CR DUP 77 = DUP IF >R .MUCH ELSE
   SWAP 76 = DUP >R IF .LITTLE ELSE
   ." JUST A SIMPLE 'M' OR 'L'? " THEN THEN DROP R> END CR CR ;



  113 LIST

( HELLO   MAIN LOOP )
: PROBELMS  ." Say! " .NAM ." I can solved all kinds of " CR
    ." problems except "
    ." those dealing with Greece. What kind of problems" CR
    ." do you have ( SEX, HEALTH, MONEY, or JOB )? "  ;
: REPLY    DUP 0 = IF .SEX ELSE DUP 1 = IF .JOB ELSE
           DUP 2 = IF .MONEY ELSE DUP 3 = IF .HEALTH ELSE
           DUP 4 = IF .MONEY ELSE THEN THEN THEN THEN THEN
           DROP   ;
: ?MORE    MORE   ANWSER CHECK  5 = ;
: HELLO    GREET  PROBLEMS  BEGIN  ANWSER  CHECK REPLY
                            ?MORE IF QUESTION AGAIN
           CR CR ." BYE."   ;
```

## HELLO, THE SECOND VERSION

As I commented in the game of ANIMALS, I was not quite satisfied with the literal translation from the BASIC game directly to FORTH, because the FORTH program had to handle the conversations explicitly. Thinking about the problem for sometime, it dawned on me that the problem of conversation might be solved simply by using the text interpreter itself to do the job. The text interpreter was built for this purpose. It is possible to define most of the expected user responses as individual words which will carry on the conversation. The program can then be simplified considerably.

Another technique used here is to store all messages on disk to be invoked by their line number using MESSAGE command.

### MESSAGE OUTPUT

| | |
|---|---|
| (LINE) | From a line number and a block number, return the address and character count of this line in buffer. |
| .LINE | Given the line and block numbers, print the line. |
| MESSAGE | Print a line relative to the beginning of Block 120. The above instructions are standard in fig-FORTH. |
| M | Abbreviation of MESSAGE. |
| KEY, EMIT | Terminal interface. |

| | |
|---|---|
| NAME | String array to store the user's name. |
| NAMED | Query the user to input his name. |
| .NAME | Print out the user's name. |
| GREET | Greeting messages and the name routine. |

### CONVERSATIONS

All the user's parts of conversation are defined as words so that the text intepreter will execute them when the user responds with these words. Instead of accepting the user's responses as answers and programing parsing and interpreting into the program, the FORTH computer accepts them as FORTH instructions to be initiated by the users.

The danger in using this program is that the users can type in the command words in any sequence out of the order intended. He can get very confused if the command word set is large.

```
  216 LIST

( HELLO, 2ND VERSION, 10-SEP-83, CHT)
: (LINE)    ( LINE# BLOCK# --- BUFFER-ADDR COUNT )
     >R     64 *    R> BLOCK    + 64    ;

: .LINE     ( LINE# BLOCK# ---  )
     (LINE)    -TRAILING    TYPE    ;

: MESSAGE   ( LINE# RELATIVE TO BLOCK 120 --- )
     16 /MOD    120 +    .LINE    ;
: M   ( N --- )    CR CR MESSAGE ;
                                                (
: KEY    0 'S 1 EXPECT ;        : EMIT    'S 1 TYPE DROP ;
: LITTLE    23 M .NAME 24 M 25 M    7 M 8 M ;
: HEALTH    9 M .NAME 10 M 11 M 12 M 13 M    7 M 8 M    ;



  217 LIST

( HELLO, GREET, 10-SEP-83, CHT)
VARIABLE NAME 38 ALLOT
: NAMED    S0 @ 40 EXPECT    0 >IN ! 0 BLK !    1 WORD COUNT
     NAME 40 BLANK    NAME SWAP MOVE    ;
: .NAME    SPACE SPACE NAME 40 -TRAILING TYPE SPACE    ;

: GREET    0 M 1 M    NAMED    2 M .NAME    3 M
     KEY 89 =    IF 4 M    ELSE 5 M .NAME 6 M THEN
     27 M .NAME 28 M 29 M 30 M    ;
: SEX    26 M  ;
: TOO    ;
: MUCH   20 M 21 M .NAME 22 M    7 M 8 M ;
: LITTLE    23 M .NAME 24 M 25 M    7 M 8 M ;
: HEALTH    9 M .NAME 10 M 11 M 12 M 13 M    7 M 8 M    ;



  218 LIST

( HELLO, 2ND VERSION, 11-SEP-83, CHT)
: JOB    14 M .NAME 15 M    16 M    7 M 8 M ;
: MONEY 17 M .NAME    18 M 19 M    7 M 8 M    ;
: YES    8 M ;
: NO     CR CR ." Bye!   " .NAME ." Have a nice day."
```

```
  120 LIST

Hello!  My name is Creating Computer.
What is your name?
Hi there!
Are you enjoying yourself here?
I am glad to hear that.
I am sorry to hear that,
, may be we can brighten your visit a bit.
Any more problems you want to solve?
What kind ( SEX, JOB, MONEY, HEALTH ) ?
My advise to you,
is:
    1. Take two tablets of aspirin.
    2. Drink plenty of fluids.
    3. Go to bed (along).
I can sympathize with you,
I have to work very long every day with no pay.


  121 LIST

My advise to you, is to open a rental computer store.
Sorry!
I am broke too.  Why don't you sell encyclopedias or marry
someone rich or stop eating, so you won't need so much money?
You call that a problem?!!  I SHOULD have that problem.
If it really bothers you,
, take a cold shower.
Why are you here!
, you should be in Tokyo or New York or Amsterdam or some place
with some action.
Is your problem TOO MUCH or TOO LITTLE ( M/L )?
Say!
, I can solved all kinds of problems except those dealing
with Greece.  What kind of problems do you have
( SEX, HEALTH, MONEY, or JOB )?
Just a simple 'YES' or 'NO' please,


  122 LIST
```