MENU: A Menu Creating Word

Contributed by Kevin Appert, Lockheed Missiles & Space Co., Inc.,
Space Sciences Laboratory, 3251 Hanover Street, Palo Alto, CA. 94304

Abstract

A technique for generating a menu from a list of FORTH words is presented.

The usual interface that a user encounters in a FORTH application is an `OK` prompt. The user is expected to type a FORTH word to get what he wants from the application. This is very often not possible for someone who is not a professional FORTH programmer. One simple solution is to provide a list of the FORTH words in the application from which the user can select the one he wants (still by typing it in). As a step beyond this, the application's vocabulary can be sealed so that the user is protected from using any of the rest of the FORTH words by mistake.

The word `MENU:` is a slightly more elegant approach. At run time it prints a list of FORTH words, hopefully with very descriptive names. The user moves the cursor to his choice and then presses the return key to execute the chosen word.

The code in the accompanying screens was implemented on a PDP-11 running FIG-FORTH. The terminal used was a VT-100 with ANSI standard escape sequences for cursor positioning.

The word `PAGE` clears the screen. The definition of `ARROW` would have to be changed to accommodate other terminals. Also, it would be easy to change the definition of `MENU:` to accommodate a terminal that does not provide absolute cursor positioning.

One possible extension would involve the use of an alternate key, such as Escape or Help, to select not the execution of the word, but a short description of what it will do when selected.

References

Laxen, Henry "Screen-Oriented Editor in FORTH", *Dr. Dobb's Journal*, Sept. 1981, p. 27.
Eaker, Charles Dr. "Case Control Structure", *Forth Dimensions*, Vol. II, No. 3, p. 37.
Laxen, Henry "Defining Words II" *Forth Dimensions*, Vol. IV, No. 2, p. 20.

Manuscript received September 1984.

```
SCR # 9                                                      KLA 07/08/83
\ case
: case ?comp csp @ ! csp 4 ; immediate

: of     4 ?pairs compile over compile = compile 0branch
    here 0 , compile drop 5 ; immediate

: endof  5 ?pairs compile branch here 0 ,
    swap 2 [compile] endif 4 ; immediate

: endcase 4 ?pairs compile drop begin sp@ csp @ = 0=
    while 2 [compile] endif repeat csp ! ; immediate

\ THIS IS DR. EAKER'S CASE STATEMENT FROM F.D.II/3p37




SCR # 10                                                     KLA 07/08/83
\ CRTXY KEY

: CRTXY ( X-POSITION Y-POSITION --- )
        SEDITOR CRTXY ;         FORTH
\ CRTXY IS AN ALIAS FOR THE CURSOR POSITIONING WORD IN THE
\ SCREEN EDITOR VOCABULARY
\ SAME AS IN HENRY LAXEN'S SCREEN EDITOR IN D.D.J. #59

: KEY SKEY ;
\ IN PDP-11 FIG FORTH, THE WORD KEY READS IN A WHOLE LINE
\ AND NOT A SINGLE KEY, SO IT HAS TO BE REPLACED WITH SKEY
\ WHICH GETS A SINGLE KEYSTROKE

: 2+ 2 + ;



SCR # 11                                                     KLA 06/29/83
\ VARIABLES

0 VARIABLE MENU.COUNT
\ MENU.COUNT WILL HOLD THE NUMBER OF ENTRIES IN THE MENU
\ CURRENTLY DISPLAYED

0 VARIABLE MENU.POINTER
\ KEEPS TRACK OF THE LINE THAT THE CURSOR IS ON

0 VARIABLE QUIT.FLAG
\ SET INSIDE THE CASE STATEMENT IN SELECT TO INDICATE THAT
\ A CHOICE HAS BEEN MADE
```

```
SCR # 12                                                   KLA 06/29/83
\ ?POINT ARROW
: ?POINT ( --- )
       MENU.POINTER @ DUP 1 < SWAP MENU.COUNT @ > OR
       IF ( OUT OF BOUNDS ) 1 MENU.POINTER ! 5 1 CRTXY THEN ;
\ ?POINT CHECKS TO SEE IF THE USER HAS WALKED THE CURSOR PAST
\ THE BEGINNING OR END OF THE MENU AND IF SO PUTS HIM BACK AT
\ THE FIRST CHOICE

: ARROW ( --- )        KEY KEY SWAP
        91 = IF ( COULD BE AN ARROW )
                CASE 66 OF 27 EMIT 91 EMIT 66 EMIT ( DOWN)
                            1 MENU.POINTER +! ?POINT ENDOF
                     65 OF 27 EMIT 91 EMIT 65 EMIT ( UP)
                 -1 MENU.POINTER +! ?POINT ENDOF
                 BEEP ( NOT UP OR DOWN ) ENDCASE THEN ;
\ ARROW CHANGES CURSOR POSITION AND MENU.POINTER

SCR # 13                                                   KLA 07/08/83
\ MENU-LIST
: MENU-LIST ( FIRST-ADDRESS --- )
     PAGE ." USE UP AND DOWN ARROWS TO MOVE CURSOR AND RETURN
TO SELECT" CR   0 MENU.COUNT !
     BEGIN DUP @ DUP
         [ ' ;S CFA ]  LITERAL = IF   1  ( FOUND ;S , SET END)
                ELSE  5 SPACES  2+ NFA  ID.  CR
                      1 MENU.COUNT +!
                      2+ ( POINT AT NEXT ENTRY IN MENU )
                      0 ( HAVE NOT FOUND ;S YET SO GO ON)
                THEN
     UNTIL  DROP DROP          5 1 CRTXY
     1 MENU.POINTER !  ( PUT CURSOR AND POINTER AT START)  ;

\ LIST THE MENU , COUNT THE ENTRIES , THEN HOME THE CURSOR


SCR # 14                                                   KLA 06/30/83
\ SELECT
: SELECT ( FIRST-ADDRESS --- )     0 QUIT.FLAG !
       BEGIN KEY
             CASE 13 OF  1 QUIT.FLAG !  ENDOF ( MADE CHOICE)
                  27 OF  ARROW          ENDOF
                  BEEP  ( NOT A RETURN OR ESCAPE CHARACTER)
             ENDCASE
       QUIT.FLAG @ UNTIL
        PAGE MENU.POINTER @ 1 -2 * + @ EXECUTE  ;

\ SELECT IS A LOOP THAT MOVES THE CURSOR UP AND DOWN UNTIL A
\ CHOICE IS MADE AND THEN EXECUTES THE CHOICE




```

```
SCR # 15                                                  KLA 07/30/83
\ MENU:
: MENU: ( ---)
        <BUILDS ] SMUDGE  DOES> DUP MENU-LIST SELECT ;

\ MENU: MAKES A MENU FROM A LIST OF FORTH WORDS. AT COMPILE
\ TIME IT MAKES A LIST OF EXECUTION VECTORS. JUST LIKE A
\ FORTH COLON DEFINITION, OR HENRY LAXEN'S CASE STATEMENT
\ FROM D.D.J.#59P32 FROM WHICH IT WAS BORROWED. AT RUN-TIME
\ IT PRESENTS A LIST OF THE FORTH WORDS AND THE USER MOVES
\ THE CURSOR WITH THE UP AND DOWN KEYS AND SELECTS WITH THE
\ RETURN KEY.
```

```
SCR # 16                                                  KLA 06/28/83
\ MENU-SAMPLE
\ HERE IS A SAMPLE OF THE USE OF MENU:
MENU: MENU-SAMPLE ( --- )
        PAGE VLIST BYE ;
```

```
SCR # 17                                                  KLA 07/30/83
\ LOAD SCREEN
: MENUSTART ;
9 LOAD
10 LOAD
11 LOAD
12 LOAD
13 LOAD
14 LOAD
15 LOAD
16 LOAD
```