ACTION! Toolbox

Lightning-fast command finder

Two powerful and widely useful routines for the ACTION! programming language. These programs work on all 8-bit Atari computers of any memory size, with disk or cassette. The ACTION! cartridge from ICD/OSS is required.

Whether you're using ACTION! to build "The Wizard of Zondar" or "The Ultimate Chef's Companion," your programming toolbox will be incomplete without a procedure that removes individual words from a string you've entered—and a procedure that compares those words with a list of known words in hopes of a match.

For efficiency and versatility, the following two procedures fill the gap nicely and can easily be customized by experienced ACTION! programmers.

1: WORDFIND

This procedure strips each Word, one at a time, from String—which is a global BYTE ARRAY similar to a BASIC string variable. In the process it discards the spaces between Words, no matter how many times you pressed the [SPACEBAR].

In its first loop, Wordfind() searches String for a non-space character, incrementing the Index into the array as it goes. Upon finding one, it stores the Index value in Start. The next loop searches for a space—and the end of the Word—while continuing to increment Index.

When another space or the end of the array is found, the procedure writes the characters between Start and Index into the global Word. Since Index, too, is a global variable, calling Wordfind() again will result in the next consecutive Word. Therefore, Index must be set to 1 before each new string is examined.

2: MATCHUP

In most applications, after you isolate a single Word you'll want to check it against the commands with which your program is prepared to deal. Matchup() can help you here.

This procedure requires that each global List of commands contains only elements of the same length. For example:

Comlist1 = "EAST WEST NORTH SOUTH"
Comlist2 = "EAWENOSO"

In Comlist1 the Increment is five—meaning that a new command begins every five characters. Comlist2 has shortened those same commands to two characters. In either case, Matchup() must be called using three parameters: the potential Command to be compared, the List of known commands and the Increment of the list.

Matchup() then jumps through the list by increments, searching the first character of each command for a match. Upon finding one, it compares the remaining characters. If all the characters match, it alters the global variable Match to show where in the list the command was found.

For example, after calling Matchup(Word, Comlist1, 5) you find that Match = 6. You then know that "Word" matched the command beginning at character 6—in this case, WEST.

Matchup() will not search past either the given Increment or a space. Thus, if you call it to examine the word WESTERLY against Comlist1, Match would still equal 6. If no match is found, Match will equal 0. As a global, Match can be used in any number of procedures, but it is always reset by the next call to Matchup().

CALLER EXAMPLE

Carefully type in Listing One, TOOLBOX.AST, and store a copy to disk before you compile and run it.

The sample Caller() procedure shows you how to use Wordfind() and Matchup(). In this example, Comlist, the command list, is "DOG CAT COW MULE". When run, the program asks you to type one of the four Words in the command list. Then the program finds the Word in the command list and prints the word and its position in the string.

Kevin Sherratt is a full-time science fiction writer and part-time programmer from London, Ontario. He is currently working on an 800XL text adventure game. This is his first appearance in Antic.
lighting-fast command finder

ACTION! TOOLBOX

LISTING 1

; ACTION! TOOLBOX
; BY KEVIN SHEPPARD
; ©1988, ANTIC PUBLISHING

MODULE
BYTE Index,
Match
BYTE ARRAY Strings,
Word,
Comlist
PROC Wordfind
BYTE Start,
Counter
FOR Counter=Index TO String#0 DO
IF Strings<Index><32 THEN
EXIT
FI
Index==1
DO
Start=Index
FOR Counter=Index TO String#0 DO
IF Strings<Index><32 THEN
EXIT
FI
Index++
DO
CopyToString,Start,Index
RETURN
PROC Matchup<BYTE ARRAY Command, List BY TE Increment>
BYTE Counter1,
Match=0
FOR Counter1=1 TO ListB STEP Increment
DO
IF Command(1)=List(Counter1) THEN
Match=1
FOR Counter2=2 TO Increment
IF List(Counter1)=Counter2-1=32 THEN
EXIT
ELSEIF Command<Counter2><List<Counter1+Comlist
Match=0
EXIT
FI
RETURN
PROC Caller
Comlist="DOG CAT COW MULE"
Print("TYPE ONE OF THE FOLLOWING: ");
PrintECommand
Input EString
Index=1
Word+=1
Matchup<Word, Comlist, 4>
PrintEWord
PrintEString
RETURN

6610 PE 7001 REM SECOND SCREEN
6618 DD 7010 GRAPHICS 0: POKE 710.417? INV1#:POS
6624 ITION 22.817? INV2#:U1=1-LD1#100:U2=1-
6630 -LD2#180
6637 NP 7020 FOR YEAR=1 TO 20: YEAR=POSITION
6645 5 YEAR
6652 MU 7025 IF REIM1##"y" AND REIM2##"y" TH
6660 EN U1=1-U1#RATE1#100:GOTO 7030
6667 UV 7030 U1=U1#RATE1#100:U1-LD1#100:LDRE
6674 IN1#="y" OR LDREIN1##"y" ORLDREIN2##"y"
6681 MG 7035 IF REIM2##"y" AND REIM2##"y" TH
6688 EN U2=2-U2#RATE2#100:GOTO 7060
6695 BZ 7060 U2=U2#RATE2#100:U2=LD2#100:LDRE
6702 IN2#="y" OR LDREIN2##"y" OR LDREIN2##"y"
6709 SM 7050 ? U1##1#100:POSITION 22.YEAR: ?
6716 U2=1#100
6723 QU 7060 NEXT YEAR
6729 HE 7070 ? 17 "esc" to return to data
6736 entry "POKE 762.1"
6743 PI 7080 ? "(w) to end program"
6750 DC 7090 GET #1X; IF X=27 THEN POKE 752.0:
6757 GOTO 1000
6764 HM 7100 IF X=42 THEN GRAPHICS 0:END
6771 UA 7110 GOTO 7080
6778 KD 10000 DIM INV1##15,INV2##15,RESPONSE
6785 #15,REIM1##15,REIM2##15,LDREIN1##15,
6792 LDREIN2##15
6799 AG 10010 CLOSE #1:OPEN #1.12.0,"K:"
6806 ZA 10020 REM DEFAULT VALUES
6813 MU 10030 INV1##=1:INV2##=0:RESPONSE
6820 #15,REIM1##=1,REIM2##=1,LDREIN1##=1,
6827 LDREIN2##=1
6834 NC 10050 GOTO 7010:
6836 MU 10060 REIM2##=1:RDREIN2##=1
6843 GW 10070 LDREIN2##="n":LDREIN2##="n"
6850 VR 10099 GOTO 1000:REM START MAIN BODY

OCTOBER 1988

ANTIC SOFTWARE LIBRARY * 77