If you've used Optimized Systems Software's ACTION! language, then you probably like it as much as I do. If you haven't, read on. ACTION! is virtually as easy to program as BASIC and as powerful as assembly language. The following demonstration programs are intended to show you BASIC hackers why you should seriously consider learning ACTION!

**SPASH IN BASIC**

SPASH1 (listing 1) is a BASIC program that demonstrates artifacting in Graphics 8. It is an extension of a short program on Antic's public domain disk GRAPHICS & SOUND #1.

Type in listing 1, check it with TYPO II and SAVE a copy. When you

---

A tutorial with four demonstration programs. For BASIC programmers who want to know about the ACTION! programming language, and for ACTION! users who want to pick up some tips. The first BASIC listing will run on any Atari computer. The remaining listings are written in ACTION! and require the ACTION! cartridge. But BASIC programmers can compare these printed listings with the first listing and get some idea why the year-old ACTION! is increasingly becoming the language of choice for serious Atari programmers.

NOTE: Antic Disk subscribers can run listing 4 without ACTION! We have provided a runtime binary file. Use the "L" option from DOS for the file, SPLASH.EXE.

RUN it, use your joystick to choose a point on the GR.8 screen. Pressing the trigger puts a "splash" of lines emanating from this center to all borders. The step size between lines can be changed by simply pressing [8]. The program lets you put as many splashes on the screen as you wish before clearing to start over. It's kind of fun—no violence, no winning score, just pretty.

**SPASH IN ACTION!**

SPASH2 (listing 2) is the same program, but in ACTION!. If you look at both listings, it is easy to see which PROcedures correspond to which BASIC subroutines. That's because I made a point of keeping SPLASH2 as structured as possible within the confines of BASIC. continued on next page
A major advantage of ACTION! is that it is a structured, procedure oriented language. It is like many of the best languages for larger computers, such as Pascal. If nothing else, working with ACTION! will improve your programming style. But there is even more...

ACTION! was designed for use on microcomputers, so certain important abilities are built in and easily accessed. It is easier to PEEK and POKE. Relocating an ARRAY is so simple that I’ve redone the Operating System line plotting routine to execute twice as fast. (More about this later.)

The BASIC command POKE 710,0 in line 202 sets the background color to black on the GR.8 screen. The ACTION! equivalent is C2 = 0 at the top of Setup. This is because of the earlier declaration BYTE C2 = 710. This establishes C2 as a BYTE variable with values 0–255. More importantly, it’s placed at memory location 710 (the register for color 2). Likewise, since we have BYTE KEY = 764, the conditional KEY<255 in ACTION! is the same as the BASIC PEEK(764)<255.

If that’s all there were, it wouldn’t seem like much. But not the least of ACTION! features is that it is a compiled language. The listing of SPLASH2 is technically just the source code. It could be written on any word processor. To run it, you must first compile it. This takes less than 2 seconds. The compiled version (object code) is full-fledged 6502 machine language; the same lightning-fast code made with assembly language. With that in mind, look at the ACTION! listing. I think it’s easier to read than BASIC. And yet, it is still just about as powerful as any assembly language.

IMPROVE OS ROUTINES

If you run SPLASH2 you’d be surprised at the seeming lack of speed. The joystick moves the center point more than twice as fast, but the splash is only marginally (5%) faster. That bothered me, and I realized the answer is simply that the Plot and DrawTo procedures of ACTION! are the same OS routines accessed from BASIC.

If you tried to improve this speed in BASIC, you’d be sunk. You’d have to write extensive USR routines in assembly language. In ACTION! things are different. You can easily write specialized routines to replace what’s in the OS and gain speed.

SPLASH3 FOR SPEED

SPLASH3 (listing 3) is functionally the same as SPLASH2. However, the “splash” moves about twice as fast because I use my own routines Dot and BLine. The top portion of the program has the file I call GR8 containing these procedures. The extra speed comes from the fact that these work in GR.8 only, and do not do any error checking. That is done elsewhere in the program.

The procedure BLine is an implementation of Bresenham’s Algorithm—one of the fastest known. But the real workhorse is the short procedure Dot. It takes advantage of the way that ACTION! treats arrays. The declaration BYTE ARRAY row creates the CARDinal pointer row to the values of the array. Then the assignment row = adrow(y) makes this point to the beginning of the y-th row of the screen (see PROC Gr8()). It is then fairly easy to move to the correct byte at row(xb) and alter it appropriately using mask arrays for the correct position Xr.

A SPLASH OF COLOR

These Dot and BLine routines are fairly easily adapted to other situations. The last program SPLASH4 (listing 4) works in the 4 colors of a GR.7+ screen. My file GR7PLUS at the top has the changes needed for these procedures. Even more speed is gained since some CARDinal variables can now be replaced by faster BYTE types. The PROCedure GR7PLUS simply alters the GR.8 display list so that the graphics area becomes GR.7+.

The program SPLASH4 will let you put splashes on the screen in any of the four available colors. I’ve also made it easy to alter these. Simply press [H][L] to alter the Hue and Luminence of the current color.

In ACTION!, like any other procedure oriented language, it is very easy to use part of one program in another. There is no worry about line number compatibility. For example, you can use my files GR8 and GR7PLUS in any of your own programs. It is easy and rewarding to build up your own library of useful routines. If you’re serious about programming your Atari, then I strongly recommend that you get into ACTION!.

(Next month’s Antic will include a fast-moving ACTION! bonus game.—Antic Ed)

ACTION!
Optimized Systems Software, Inc.
19218 Kentwood Ave.
San Jose, CA 95129
(408) 446-3099
16K cartridge
$99
demo of ACTION! vs. BASIC

SPLASH IN ACTION!

LISTING 1

FI 10 REM SPLASH 1
DA 12 REM BY PAUL CHABOT
AN 14 REM ANTIC MAGAZINE
UZ 20 REM MAIN LOOP
OS 22 GOSUB 200
SQ 24 GOSUB 100:GOSUB 50
CX 26 POKE 656.3:POKE 657.2
S 28 ?" Another [C]=Clear"
EG 30 K=PEEK(764):IF K=255 THEN 30
OZ 32 POKE 764.255
LH 34 IF K=18 THEN 20
QT 36 GOTO 24
LZ 50 REM SPLASH
SA 52 POKE 712.16=INT(RND(0)*16)+2
ER 60 FOR I=0 TO 319 STEP 5
NN 62 PLOT X,Y:DRAW 1.0:PLOT X,Y
ED 64 DRAW 1.159:NEXT I
GL 66 FOR I=0 TO 319 STEP 5
HM 68 PLOT X,Y:DRAW 319.1:PLOT X,Y
KS 70 DRAW 0.1:NEXT I
AB 72 RETURN
MK 100 REM JOYSTICK
FC 102 POKE 656.3:POKE 657.2
BO 104 ?"[Trigger] = SPLASH"
';
KP 110 POKE 656.4:POKE 657.9
EX 112 ?"X:";"Y:";
YF 120 ST=STICK(0):IF STRIC(0)=0 THEN 140
NT 122 IF PEEK(764)+(255 THEN POKE 764,255
;GOSUB 150
GE 124 IF ST=15 THEN 128
VB 130 IF ST>7 AND X<319 THEN X=X+1
FO 132 IF ST=11 AND X>0 THEN X=X-1
VA 134 IF ST=13 AND Y<159 THEN Y=Y+1
KG 136 IF ST=14 AND Y>0 THEN Y=Y-1
MS 150 GOTO 110
ZF 152 REM INC STEP
OH 154 S=S+1:IF S>16 THEN S=1
WH 156 POKE 656.1:POKE 657.255? 5;"
IN 158 POKE 712.16=INT(RND(0)*16)+2
AF 160 RETURN
OK 200 REM SETUP
FB 202 GRAPHICS 8:POKE 710.0:POKE 709.14
IC 204 POKE 712.16=INT(RND(0)*16)+2
IO 206 POKE 752.1:COLOR 1:K=128:V=60:S=7
HS 210 ?"0"
EC 212 ?"CENTER 120 60 STEP 7"
WI 214 ?"[Joystick] [S]"
ZI 222 RETURN

LISTING 2

: SPLASH 2
: PAUL CHABOT
: MODULE
: BYTE C=789,E=2710,bor=712,color=752
: .key=764,trow=656,tcol=657,w=5
: CARD x
: PROC Setup()
: Graphics(8):C=8:C=14:cur=1:color=1
: bor=16*RND(16)+2:W=128:Y=60:S=7
: Print("[KB] & [JS] SPLASH"
: Print("[CENTER 120 60 STEP 7"
: Print("[Joystick] [S]"
: RETURN

LISTING 3

: SPLASH 3
: -----
: GB 8
: PAUL CHABOT
: MODULE
: BYTE ARRAY mask=128 64 32 16 8 4 2 1
: CARD ARRAY adrow(160)
: PROC Color(BYTE c)
: BYTE 1
: FOR i=0 TO 7 DO
: .mask(7:i)=c:ccc=LSH 1
: OD
: RETURN
: PROC Dot(CARD x,BYTE y)
: BYTE xy,xr
: BYTE ARRAY row
: .premask=(127 191 233 239 247 251 253 254)
: .xbus b=3:xr=x y=2:row=adrow()
: row(xy)=&premask(xy) x mask(xy)
: RETURN

ANTIC SOFTWARE LIBRARY

APRIL 1985
PROC Bline( cardi1, byte w1, card x2, byte w2 )
BYTE y, x, yf, yf,
CARD x, y
INT a, b, c, dx, dy
DOT(x1, y1): DOT(x2, y2)
IF x2 >= x1 THEN dx = x2 - x1: xf = 0
ELSE dx = x1 - x2: xf = 1
IF y2 >= y1 THEN dy = y2 - y1: yf = 0
ELSE dy = y1 - y2: yf = 1
IF dx < 0 AND dy < 0 THEN RETURN FI
y = w1: x = w2
IF dx * dy THEN adx = dy: ady = dx: adt = dy
FOR i = 2 TO dy DO
IF y + dy THEN y = y - 1 ELSE y = -1 FI
IF x + dx THEN x = x - 1 ELSE x = -1 FI
FI DOT(x, y)
OD
ELSE adx = dx: ada = dy: adt = dy
FOR i = 2 TO dy DO
IF y - dy THEN y = y + 1 ELSE y = 1 FI
IF x - dx THEN x = x + 1 ELSE x = 1 FI
FI DOT(x, y)
OD
FI
RETURN

PROC GR8()
BYTE x = 16
CARD s = 80
Graphics(b): bor = 16: adrow(0) = 8
FOR i = 0 TO 159 DO
adrow(i) = adrow(i - 1) + 8
OD
RETURN

; Variant of SPLASH

MODULE BYTE cs = 789, cs = 789, bor = 712, cur = 752,
, key = 764, trow = 656, tc01 = 657, 9, 5
CARD x

PROC Setup()
GR8(): cs = 28: cs = 14: cur = 1: x = 120: y = 60: s = 7
bor = 16: rand(150) = 2
PRINT("CEN T E R 1 2 8 , 6 0 S T E P 7 \")
PRINT("J O Y S T I C K )
\[ S 1 ]")
RETURN

PROC Splash()
CARD i
bor = 16: rand(150) = 2
FOR i = 0 TO 159 STEP 2 DO
BLINE(x, y, 0): BLINE(x, y, 1, 159)
OD
FOR i = 0 TO 159 STEP 2 DO
BLINE(x, y, 0): BLINE(x, y, 1, 159)
OD
RETURN

PROC IncStep()
set = 1: bor = 16: rand(15) = 2
IF 3 > 16 THEN 1 FI
trow = 1: tcol = 25: PRINT"(5): PRINT"("
RETURN

PROC Josstick()
BYTE st
TROUT c3 = 2
PRINT("TRIGGER") SPLASH "")
DO trow = 1: tcol = 0: st = stick(0)
PRINT(0: PRINT"(": PRINT(0: PRINT"("
WHILE st = 15 DO
IF STRI(0) = 0 THEN RETURN FI
IF key = 255 THEN key = 255: IncStep(1) FI
OD
PRINT(0: PRINT(0: PRINT"("
IF st = 1 AND x = 319 THEN x = 1
ELSEF st = 1 AND y = 8 THEN x = 1
ELSEF st = 15 AND y = 159 THEN y = 1
ELSEF st = 14 AND y = 8 THEN y = 1
FI
RETURN

PROC Main()
DO key = 255: Setup0
DO Josstick(): SPLASH()
Drow(3): tcol = 3
PRINT("A-MUTHER, I-CLEAN")
WHILE key = 255 DO
IF key = 16 THEN EXIT FI
key = 255
OD
OD
RETURN

LISTING 4

; SPLASH

; Paul Chabot

; MODULE ARRAY MASK = (64, 16, 1)
CARD adrow(16)

PROC Color( byte c )
MASK(0) = MSK Bắc (2) = LSH 2
MASK(1) = LSH 4 : MASK(8) = LSH 6
RETURN

PROC Dot( byte x, y )
BYTE x, y
BYTE ARRAY row,
, premask = (63, 257, 245, 251)
X = RSH 2: Y = RSH AND STROW = adrow(y)
ROW(0) = 0: PREMASK (y) = MASK(y)
RETURN

PROC Bline( card1, card2, card3, card4 )
BYTE x, y, xf, yf,
INT a, b, c, dx, dy
DOT(x1, y1): DOT(x2, y2)
IF x2 >= x1 THEN dx = x2 - x1: xf = 0
ELSE dx = x1 - x2: xf = 1
IF y2 >= y1 THEN dy = y2 - y1: yf = 0
ELSE dy = y1 - y2: yf = 1
IF dx < 0 AND dy < 0 THEN RETURN FI
x = w1: y = w2
IF dx * dy THEN adx = dy: ady = dx: adt = dy
FOR i = 2 TO dy DO
IF y + dy THEN y = y - 1 ELSE y = -1 FI
IF x + dx THEN x = x - 1 ELSE x = -1 FI
FI DOT(x, y)
OD
ELSE adx = dx: ada = dy: adt = dy
FOR i = 2 TO dy DO
IF y - dy THEN y = y + 1 ELSE y = 1 FI
IF x - dx THEN x = x + 1 ELSE x = 1 FI
FI DOT(x, y)
OD
FI
RETURN

PROC GR8()
## TABLE X

### TAX SPREADSHEET

### UPDATE

**syncal tax preparation follow-up**

**Article on page 34**

### TABLE Y

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>0.08</td>
</tr>
<tr>
<td>84</td>
<td>5.900</td>
</tr>
<tr>
<td>85</td>
<td>0.11</td>
</tr>
</tbody>
</table>

**APRIL 1985**