SPLASH in ACTION!

Demo of Action! vs. Basic

by PAUL CHABOT

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.

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!

SPLASH IN BASIC

SPLASH1 (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 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 [S]. 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. . .

SPLASH IN ACTION!

SPLASH2 (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 40 bytes 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

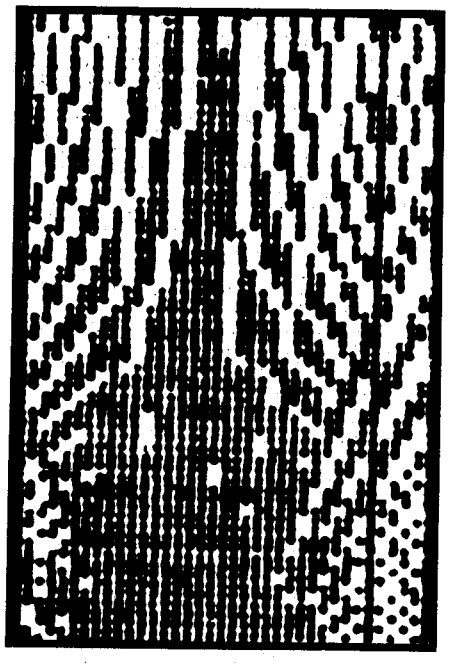
These **Dot** and **BLine** routines are fairly easily adapted to other situations. The last program SPLASH4 (listing 4) works in the 4 colors of a GR.7+ screen. My file GR7PLUS at the top has the changes needed for these procedures. Even more speed is gained since some CARDinal variables can now be 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.
1221B Kentwood Ave.
San Jose CA, 95129
(408) 446-3099
16K cartridge
$99



Professor Paul Chabot teaches in the Mathematics and Computer Science Department at California State University, Los Angeles.

Listing on page 70.

SPLASH IN ACTION!

Article on page 43.

LISTING 1

```
FI 10 REM SPLASH 1
DA 12 REM BY PAUL CHABOT
RN 14 REM ANTIC MAGAZINE
UZ 20 REM MAIN LOOP
OS 22 GOSUB 200
SO 24 GOSUB 100:GOSUB 50
CX 26 POKE 656,3:POKE 657,2
SX 28 ? "[A]-Another          [C]-Clear";
GK 30 K=PEEK(764):IF K=255 THEN 30
OZ 32 POKE 764,255
LH 34 IF K=18 THEN 20
TT 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:DRAWTO I,0:PLOT X,Y
EB 64 DRAWTO I,159:NEXT I
GL 66 FOR I=0 TO 159 STEP 5
HM 68 PLOT X,Y:DRAWTO 319,I:PLOT X,Y
KS 70 DRAWTO 0,I:NEXT I
AB 72 RETURN
MK 100 REM JOYSTICK
FC 102 POKE 656,3:POKE 657,2
BO 104 ? "[trigger] - SPLASH
       ";
KP 110 POKE 656,1:POKE 657,9
XE 112 ? X;" , ";Y;"  ";
YF 120 ST=STICK(0):IF STRIG(0)=0 THEN 140
NT 122 IF PEEK(764)<255 THEN POKE 764,255
   :GOSUB 150
QE 124 IF ST=15 THEN 120
YB 130 IF ST=7 AND X<319 THEN X=X+1
FO 132 IF ST=11 AND X>0 THEN X=X-1
VH 134 IF ST=13 AND Y<159 THEN Y=Y+1
KG 136 IF ST=14 AND Y>0 THEN Y=Y-1
MV 138 GOTO 110
ZF 140 RETURN
OW 150 REM INC STEP
NX 152 S=S+1:IF S>16 THEN S=1
VW 154 POKE 656,1:POKE 657,25:? S;" ";
IR 156 POKE 712,16*INT(RND(0)*16)+2
AF 158 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
IQ 206 POKE 752,1:COLOR 1:X=120:Y=60:S=7
WS 210 ? "   GR.8          S P L A S H
   "
EC 212 ? "CENTER 120 , 60   STEP 7  "
WK 214 ? "     [joystick]       [S] "
ZI 222 RETURN
```

LISTING 2

```
;   SPLASH 2
; Paul Chabot
;
MODULE
BYTE c1=709,c2=710,bor=712,cur=752
  ,key=764,trow=656,tcol=657,y,s
CARD x

PROC Setup()
Graphics(8):c2=0:c1=14:cur=1:color=1
bor=16*Rand(16)+2:x=120:y=60:s=7
PrintE("   GR.8          S P L A S H          ")
PrintE("CENTER 120 , 60   STEP 7 ")
PrintE("     [joystick]       [S]   ")
RETURN

PROC Splash()
CARD i
bor=16*Rand(16)+2
FOR i=0 TO 319 STEP s DO
  Plot(x,y):DrawTo(i,0)
  Plot(x,y):DrawTo(i,159)
OD
FOR i=0 TO 159 STEP s DO
  Plot(x,y):DrawTo(319,i)
  Plot(x,y):DrawTo(0,i)
OD
RETURN

PROC IncStep()
s==+1:bor=16*Rand(16)+2
IF s>16 THEN s=1 FI
trow=1:tcol=25:PrintB(s):Print(" ")
RETURN

PROC Joystick()
BYTE st
trow=3:tcol=2
Print("[trigger] - SPLASH        ")
DO trow=1:tcol=9:st=Stick(0)
  PrintC(x):Print(" , "):PrintB(y):Print("  ")
  WHILE Stick(0)=15 DO
    IF Strig(0)=0 THEN RETURN FI
    IF key<255 THEN key=255:IncStep() FI
  OD st=Stick(0)
  IF st=7 AND x<319 THEN x==+1
  ELSEIF st=11 AND x>0 THEN x==-1
  ELSEIF st=13 AND y<159 THEN y==+1
  ELSEIF st=14 AND y>0 THEN y==-1
  FI
OD
RETURN

PROC Main()
DO key=255:Setup()
  DO Joystick():Splash()
    trow=3:tcol=2
    Print("[A]-Another      [C]-Clear")
    WHILE key=255 DO OD
      IF key=18 THEN EXIT FI
    key=255
  OD
OD
RETURN
```

LISTING 3

```
;  SPLASH 3
;--------------------------------------
;  Gr8
; Paul Chabot
;
MODULE
BYTE ARRAY mask=[128 64 32 16 8 4 2 1]
CARD ARRAY adrow(160)

PROC Clr(BYTE c)
BYTE i
FOR i=0 TO 7 DO
  mask(7-i)=c:c==LSH 1
OD
RETURN

PROC Dot(CARD x,BYTE y)
BYTE xb,xr
BYTE ARRAY row
  ,premask=[127 191 223 239 247 251 253 254]
xb=x RSH 3:xr=x AND 7:row=adrow(y)
row(xb)==& premask(xr) % mask(xr)
RETURN
```

```
PROC BLine(CARD x1,BYTE y1,CARD x2,BYTE y2)
BYTE y,xf,yf,j
CARD x,i
INT a,b,t,dx,dy
Dot(x1,y1):Dot(x2,y2)
IF x2>x1 THEN dx=x2-x1:xf=0
ELSE dx=x1-x2:xf=1 FI
IF y2>y1 THEN dy=y2-y1:yf=0
ELSE dy=y1-y2:yf=1 FI
IF dx<2 AND dy<2 THEN RETURN FI
x=x1:y=y1
IF dx>dy THEN a=dy+dy:t=a-dx:b=t-dx
  FOR i=2 TO dx DO
    IF xf=0 THEN x==+1 ELSE x==-1 FI
    IF t<0 THEN t==+a
    ELSE t==+b
      IF yf=0 THEN y==+1 ELSE y==-1 FI
    FI Dot(x,y)
  OD
ELSE a=dx+dx:t=a-dy:b=t-dy
  FOR j=2 TO dy DO
    IF yf=0 THEN y==+1 ELSE y==-1 FI
    IF t<0 THEN t==+a
    ELSE t==+b
      IF xf=0 THEN x==+1 ELSE x==-1 FI
    FI Dot(x,y)
  OD
FI
RETURN

PROC Gr8()
BYTE bor=710,i
CARD sa=88
Graphics(8):bor=18:adrow(0)=sa
FOR i=1 TO 159 DO
  adrow(i)=adrow(i-1)+40
OD
RETURN
;-------------------------------------
; Variant of SPLASH
;
MODULE
BYTE c1=709,c2=710,bor=712,cur=752
  ,key=764,trow=656,tcol=657,y,s
CARD x

PROC Setup()
Gr8():c2=0:c1=14:cur=1:x=120:y=60:s=7
bor=16*Rand(16)+2
PrintE("  GR.8          S P L A S H          ")
PrintE("CENTER 120 , 60    STEP 7 ")
PrintE("      [joystick]       [5] ")
RETURN

PROC Splash()
CARD i
bor=16*Rand(16)+2
FOR i=0 TO 319 STEP s DO
  BLine(x,y,i,0):BLine(x,y,i,159)
OD
FOR i=0 TO 159 STEP s DO
  BLine(x,y,0,i):BLine(x,y,319,i)
OD
RETURN

PROC IncStep()
s==+1:bor=16*Rand(16)+2
IF s>16 THEN s=1 FI
trow=1:tcol=25:PrintB(s):Print(" ")
RETURN

PROC Joystick()
BYTE st
trow=3:tcol=2
Print("[trigger] - SPLASH        ")
DO trow=1:tcol=9:st=Stick(0)
  PrintC(x):Print(" , "):PrintB(y):Print("  ")
  WHILE Stick(0)=15 DO
    IF Strig(0)=0 THEN RETURN FI
    IF key<255 THEN key=255:IncStep() FI
  OD st=Stick(0)
  IF st=7 AND x<319 THEN x==+1
  ELSEIF st=11 AND x>0 THEN x==-1
  ELSEIF st=13 AND y<159 THEN y==+1
  ELSEIF st=14 AND y>0 THEN y==-1
  FI
OD
RETURN

PROC Main()
DO key=255:Setup()
  DO Joystick():Splash()
    trow=3:tcol=2
    Print("[A]-Another      [C]-Clear")
    WHILE key=255 DO OD
      IF key=18 THEN EXIT FI
    key=255
  OD
OD
RETURN
```

LISTING 4

```
;  SPLASH4
;-------------------------------------
;  Gr7plus
; Paul Chabot
;
MODULE
BYTE ARRAY mask=[64 16 4 1]
CARD ARRAY adrow(160)

PROC Clor(BYTE c)
mask(3)=c:mask(2)=c LSH 2
mask(1)=c LSH 4:mask(0)=c LSH 6
RETURN

PROC Dot(BYTE x,y)
BYTE xb,xr
BYTE ARRAY row
  ,premask=[63 207 243 252]
xb=x RSH 2:xr=x AND 3:row=adrow(y)
row(xb)==& premask(xr) % mask(xr)
RETURN

PROC BLine(BYTE x1,y1,x2,y2)
BYTE x,y,xf,yf,i
INT a,b,t,dx,dy
Dot(x1,y1):Dot(x2,y2)
IF x2>x1 THEN dx=x2-x1:xf=0
ELSE dx=x1-x2:xf=1 FI
IF y2>y1 THEN dy=y2-y1:yf=0
ELSE dy=y1-y2:yf=1 FI
IF dx<2 AND dy<2 THEN RETURN FI
x=x1:y=y1
IF dx>dy THEN a=dy+dy:t=a-dx:b=t-dx
  FOR i=2 TO dx DO
    IF xf=0 THEN x==+1 ELSE x==-1 FI
    IF t<0 THEN t==+a
    ELSE t==+b
      IF yf=0 THEN y==+1 ELSE y==-1 FI
    FI Dot(x,y)
  OD
ELSE a=dx+dx:t=a-dy:b=t-dy
  FOR i=2 TO dy DO
    IF yf=0 THEN y==+1 ELSE y==-1 FI
    IF t<0 THEN t==+a
    ELSE t==+b
      IF xf=0 THEN x==+1 ELSE x==-1 FI
    FI Dot(x,y)
  OD
FI
RETURN

PROC Gr7plus()
BYTE i
BYTE ARRAY dl
CARD sa=88,dlist=560
Graphics(8):adrow(0)=sa
FOR i=1 TO 159 DO
  adrow(i)=adrow(i-1)+40
OD
dl=dlist:dl(3)=78:dl(99)=78
FOR i=6 TO 98 DO dl(i)=14 OD
FOR i=102 TO 166 DO dl(i)=14 OD
RETURN
;-------------------------------------
; COLOR SPLASH
;
MODULE
BYTE cur=752,key=764,trow=656,tcol=657
  ,x,y,s,c,i,j
BYTE ARRAY creg=708
```

continued on next page

```
 ,dfault=[54 26 194 0 80]

PROC Splash()
FOR i=0 TO 159 STEP s DO
  BLine(x,y,i,0):BLine(x,y,i,159)
  BLine(x,y,0,i):BLine(x,y,159,i)
OD
RETURN

PROC IncStep()
s==+1:IF s>16 THEN s=1 FI
trow=1:tcol=26:PrintB(s):Print(" ")
RETURN

PROC IncColor()
i=c:c==+1
IF c>3 THEN c=0:i=4 FI
Clor(c):i=creg(i)
trow=1:tcol=37:PrintB(c):Print(" ")
trow=2:tcol=36:PrintB(i RSH 4):Print(" ")
trow=3:tcol=36:PrintB(i & 14):Print(" ")
RETURN

PROC IncHue()
IF c=0 THEN i=4 ELSE i=c-1 FI
j=creg(i) RSH 4:j==+1
IF j>15 THEN j=0 FI
trow=2:tcol=36:PrintB(j):Print(" ")
creg(i)=(j LSH 4)+(creg(i) & 14)
RETURN

PROC IncLum()
IF c=0 THEN i=4 ELSE i=c-1 FI
j=creg(i) & 14:j==+2
IF j>15 THEN j=0 FI
trow=3:tcol=36:PrintB(j):Print(" ")
creg(i)=(creg(i) & 240)+j
RETURN

PROC Joystick()
BYTE st,k
DO trow=1:tcol=9
 PrintC(x):Print(" , "):PrintB(y):Print(" ")
 WHILE Stick(0)=15 DO
   IF Strig(0)=0 THEN Splash() FI
   IF key<255 THEN k=key:key=255
     IF k=62 THEN IncStep()        ;S
     ELSEIF k=18 THEN IncColor()   ;C
     ELSEIF k=57 THEN IncHue()     ;H
     ELSEIF k=0 THEN IncLum()      ;L
     ELSEIF k=35 THEN RETURN       ;N
     FI
   FI
 OD st=Stick(0)
 IF st=7 AND x<159 THEN x==+1
 ELSEIF st=11 AND x>0 THEN x==-1
 ELSEIF st=13 AND y<159 THEN y==+1
 ELSEIF st=14 AND y>0 THEN y==-1
 FI
OD
RETURN

PROC Setup()
Gr7Plus():cur=1
FOR i=0 TO 4 DO creg(i)=dfault(i) OD
PrintE("  Gr7Plus          S P L A S H           ")
PrintE("CENTER 80 , 60    [S]tep 7   [C]OLOR")
PrintE("     [Joystick]                 [H]ue")
Print("[trig]-SPLASH  [N]ew Screen [L]um")
x=80:y=60:s=7:c=0:IncColor()
RETURN

PROC OpenScene()
Setup():x=20:y=20:s=9:Splash()
IncColor():x=50:y=110:s=7:Splash()
IncColor():x=120:y=60:s=9:Splash()
IncColor():x=80:y=130:s=9:Splash()
IncColor():x=140:y=130:s=7:Splash()
RETURN

PROC Main()
OpenScene():Joystick()
DO Setup():Joystick() OD
RETURN
```

syncalc tax preparation follow-up!

84 TAX SPREADSHEET UPDATE

Article on page 34.

TABLE X

	A	B	C
66	SCHEDULE	X SINGLE	
67	2,300	0	0.11
68	3,400	121	0.12
69	4,400	241	0.14
70	6,500	535	0.15
71	8,500	835	0.16
72	10,800	1,203	0.18
73	12,900	1,581	0.20
74	15,000	2,001	0.23
75	18,200	2,737	0.26
76	23,500	4,115	0.30
77	28,800	5,705	0.34
78	34,100	7,507	0.38
79	41,500	10,319	0.42
80	55,300	16,115	0.48
81	81,800	28,835	0.50

TABLE Y

	A	B	C
82	SCHEDULE	Y MARRIED	
83	1	0	0.00
84	3,400	0	0.11
85	5,500	231	0.12
86	7,600	483	0.14
87	11,900	1,085	0.16
88	16,000	1,741	0.18
89	20,200	2,497	0.22
90	24,600	3,465	0.25
91	29,900	4,790	0.28
92	35,200	6,274	0.33
93	45,800	9,772	0.38
94	60,000	15,168	0.42
95	85,600	25,920	0.45
96	109,400	36,630	0.49
97	162,400	62,600	0.50