Pulse in Action!

16K Cassette or 24K Disk

by Joel Gluck

It’s not easy to write an article to accompany a simple graphic demo, and it’s probably just as tough to read one. But before you race to your computer, plug in your Action! cartridge, type in Pulse and run it, why not take a few seconds to read what it’s taken a few hours for me to write?

For starters, I’ll admit that what Pulse does is beautiful. The program draws several brightly-colored horizontal lines on the screen and makes them expand and contract at various rates. This sounds simple, but the combinations are almost infinite—and frequently complex. When I first wrote Pulse and ran it, I shut off the room lights and spent a good long time staring at the screen. No, the effect is not hypnotic, and, no, I wasn’t on drugs at the time—the fact is that, as a human being, I appreciate beauty, and Pulse gave me a sizable dose.

Despite its beauty, however, Pulse is not a work of art. To me, a work of art must relate to the human experience, and Pulse is merely a random and abstract visual creation. It does not affect me deeply, the way a good novel, play or piece of music can. The best way to describe it is “emptiness”—Pulse is empty. Rated as art, it is bad art.

I’m not saying that I’m going to stop fooling around with graphic demos on my Atari 800. All I’m saying is that there is more to art than beauty or simple emotional effects, and this fact is a challenge to myself and to all creators of “computer art.” Translation: I’m still thinking. I hope you are, too.

Using the program.
Plug in an Action! cartridge and type the source code as written into the editor. Save it to disk or tape and then run.

Pulse is a simple graphic demo. For the greatest effect, run the program with all room lights extinguished. Press RETURN to exit the program. Press any other keyboard key for new patterns. The program will automatically display a new pattern every 15-20 seconds.

Action! listing.

; Pulse - Joel Gluck - Analog
BYTE ARRAY x(192), y(192), c(192)
BYTE NUM=100
INT ARRAY xld(192)
CARD ARRAY linept(192)

PROC pauz(CARD n)
CARD i
FOR i=0 TO n+n DO OD
RETURN

PROC intro()
BYTE i, COLOR1=709
Graphics(8)
Poke(710, 0)
Poke(752, 1)
Print"Welcome"
Pauz(30000)
Position(17, 9)
Print("Pulse")
pause(30000)
position(13,11)
Print("by Joel Gluck")
pause(30000)
position(9,13)
Print("from ANALOG COMPUTING")
pause(30000)
FOR i=0 TO 15 DO
COLOR=15-i
pause(2000)
OD
RETURN

PROC gridinit()
CARD line,reg,col,lum,scrn=88
BYTE ARRAY gtiacol(0)=705
Graphics(10)
POKE(784,0)
FOR reg=0 TO 7 DO
col=Rand(16)+
lum=Rand(5)+4
gtiacol(reg)=col*lum+1+lum
OD
FOR line = 0 TO 191 DO
linept(line)=scrn+40*line
OD
RETURN

PROC plot1o(BYTE x,y,col)
BYTE POINTER pixel
BYTE ARRAY
colfil= [0 17 34 51 68 85 102 119 136 0 0 0 0 0 0 0]
mask= [15 240]
mask2= [124 153]
pixel=linept(y)+x RSH 1
pixel^=pixel ^ & mask(x & 1)
% (colfil[col] & mask2(x & 1))
RETURN

BYTE FUNC locate1o(BYTE x,y)
BYTE POINTER pixel
BYTE ARRAY mask= [240 153]
pixel=linept(y)+x RSH 1
RETURN(pixel & mask(x & 1)) RSH ((x & 1) XOR 1) LSH 1)

PROC drawline(BYTE a,b,c)
BYTE i
FOR i=0 TO 79-a DO
plot10(i,b,c)
plot10(i,191-b,c)
OD
RETURN

PROC init()
BYTE i,j,s
gridinit()
FOR i=0 TO num-1 DO
x(i)=Rand(48)
DO
y(i)=Rand(96)
S=0
IF i=0 THEN
EXIT
FI
FOR j=0 TO i-1 DO
IF y(i)=y(j) THEN
S=0
EXIT
FI
OD
UNTIL S=0
OD
c(i)=(i MOD 8)+1
drawline(x(i),y(i),c(i))

xd(i)=(Rand(2)*2-1)*(Rand(3)+1)
OD
RETURN

PROC squeeze(BYTE n)
BYTE a,d
INT d

d=xd(n)
IF d<0 THEN
d=-1
ELSE
d=1
FOR i=1 TO (xd(n)/d) DO
a=x(n)+d
IF a<59 THEN
xd(n)=xd(n)
EXIT
FI
IF d>0 THEN
plot10(x(n),y(n),0)
plot10(79-x(n),y(n),0)
plot10(x(n),191-y(n),0)
plot10(79-x(n),191-y(n),0)
ELSE
plot10(x(n),y(n),c(n))
plot10(79-x(n),y(n),c(n))
plot10(x(n),191-y(n),c(n))
plot10(79-x(n),191-y(n),c(n))
FI
x(n)=a
plot10(x(n),y(n),c(n))
plot10(79-x(n),y(n),c(n))
plot10(x(n),191-y(n),c(n))
plot10(79-x(n),191-y(n),c(n))
OD
RETURN

PROC pulse()
BYTE i,CH=764,AT=77,MS=19,LS=20
introl()
DO
num=Rand(6)+5
init()
CH=255
AT=77
LS=0
MS=0
DO
FOR i=0 TO num-1 DO
squeeze(i)
OD
UNTIL CH<>255 OR MS<4
OD
UNTIL CH=12
OD
CH=255
Graphics(0)
RETURN

---

Next issue: **Bounce in Action!**