Trails in Action!

Mesmerizing graphic fun with your *KoalaPad* or joystick.

by Kevin R. Garlow

After receiving an Action! cartridge and a **KoalaPad** for Christmas, I set out to learn the language and make use of the **KoalaPad** at the same time. I began by typing in the kaleidoscope program by Clinton Parker in **ANALOG Computing**'s issue 17. First, I developed this into a sketching program that used the **KoalaPad**. A day later, I remembered a demo program I'd once seen run on the Macintosh. I set out to duplicate that program based on my recollection of it, using the **KoalaPad** in place of the mouse. The result is **Trails**.

Trails is an interactive graphics demo written in Optimized Systems Software's Action! language. To use it, you need the Action! cartridge (for those without Action!, a runtime version of this program may be found on this month's disk, or on the Delphi Atari Users' Group.) A **KoalaPad** is highly recommended, but alternate procedures are provided to let you use a joystick.

Typing it in.

Type in Listing 1, then verify your work with **D:CHECK in Action!** from issue 44. If you'll be using a joystick with the program, you must also type in Listings 2 and 3, then replace the Instructions() procedure and the Draw() procedure in Listing 1 with Listings 2 and 3, as appropriate.

Directions.

Trails allows you to draw on the graphics 8 screen, via **KoalaPad** or joystick. After approximately five seconds, the points you drew will begin to be erased, in the order you drew them (meanwhile, you can continue to draw). The effect can be hypnotic, especially when the mirror function is on.

Your drawing is made symmetrical by the mirroring effect. Drawing in the center produces a pleasing lace effect, while drawing on the edges and in the corners gives you interestingly symmetrical loops and lines. Fascinating patterns emerge as you race to the center, out to the edges, then back to the center again, while the computer sequentially "undraws" what you've done.

The left button of the **KoalaPad** clears the screen. The right button both clears the screen and toggles the mirror effect on and off. With the mirror off, you can experiment with "chasing your tail," or write your name in cursive and watch it be unwritten.

While the program runs, an additional option is open to you. By hitting any key on the keyboard (except BREAK, which stops the program), you're shown the instructions and told the current value of persistence. You may then enter a new value from *1* to *10*. The persistence controls the "lag time" between when a dot is drawn and when it's erased. As the program stands, the number you enter is approximately the lag time in seconds. If you don't wish to change the persistence, you can simply hit RETURN. Entering *0* or a letter will return you to the Action! monitor.

Alternate procedures are given, for joystick use. They replace the Draw() and Instructions() procedures in the original. The joystick button has the same action as the right button of the **KoalaPad**—it clears the screen and turns the mirror mode on or off. You can still change the persistence value by hitting a key.

I didn't make the joystick an option within the program, because I believe the **KoalaPad** works better as an input device in this program. You should use the original procedures if you have a **KoalaPad**.

Explanation.

The Main() procedure first calls the procedure Instructions(), which explains how to use the program. Then it enters a loop which first calls Setup(mode). This clears the screen, zeroes the *x* and *y* arrays, and draws a box on the screen. Main() then calls Draw(), the heart of the program. Draw() reads the **KoalaPad** (or joystick) and modifies x0 and y0 accordingly. The first call to Octplot() erases the point that occurred maxpts before the present point. Then the current point is saved in the x and y arrays, for erasure later on.

Finally, the present point is plotted. The variable n keeps track of the current position in the arrays. If a button is pushed, the Draw() routine calls Setup(mode) again to clear the screen. Hitting a key will return to Main(), which will call Get_pers(). Get_pers() calls Instructions() then displays the current value of persistence (maxpts). This is where you can enter a new value. If a *0* is entered, you are returned to Main() and then to the Action! monitor.

Modifications.

There are a number of modifications to this program that come to mind. One of the easiest potential changes occurs at the beginning of the program listing. The constant pers_ratio could be DEFINEd to be a lesser or greater integer than *35* (as it is in the listing). A larger number would extend the lag time across the entire range of persistence values of *1* to *10*, while a smaller number would decrease the lag time. Please note that, if you change pers_ratio to a number greater than *50*, you need to dimension the x and y arrays to accommodate the additional points. For example, if you DEFINE pers_ratio to be *65*, you should dimension x and y to *650* (i.e., *CARD ARRAY x(650), y(650)*).

Another modification might be to add more color. For example, change the program to run in graphics 7.5, the high-resolution four-color graphics mode. Or how about plotting four points in a square each time you plot a position? This would be fairly easy to do, by having Draw() call a new routine, say Quadplot(), that would in turn call Octplot() four times. And to speed things up, you could use the quick plotting routines from Clinton Parker's article in **ANALOG Computing**'s issue 18. Let your imagination run free and match the program to your own idea of how it should be.

Summary.

Trails is an intriguing demo, because it's both kinetic and interactive. It must be seen in action (so to speak) to be appreciated. A still picture couldn't do it justice. At the same time, it requires someone to do the drawing. By itself, it sits and repeatedly plots and unplots the same points. With someone using the program, it draws mesmerizing patterns on the screen. Enjoy!

Kevin R. Garlow is majoring in Astronomy and Astrophysics at Villanova University. He's owned an Atari 800 for two and a half years, and uses it for word processing, programming in BASIC (and some assembly) and, of course, playing games. His favorite ***ANALOG Computing*** *games are* ***Bacterion!*** *and* ***Planetary Defense****.*

Listing 1.
Action! listing.

```
;TRAILS.ACT
; Copyright 1986 by Kevin Garlow
; Last Modified 1-3-86

;      CHECKSUM DATA
;[E7 5F 32 D9 1B 77 3D 89
; 74 05 6C C4 EB C2 6B ]

DEFINE pers_ratio = "35"

BYTE col1=709,col2=710,border=712,
     cur=752,key=764,attract=77,
     clock=20,mirror=[1],mode
CARD ARRAY  x(500),y(500)
CARD x0,y0,n=[0],i,maxpts=[175]


PROC Setup(BYTE mode)

FOR i=0 TO maxpts DO
  x(i)=0:y(i)=0
OD
Graphics(mode) : col2=0 : col1=14
cur=0 : color=1 : border=16*Rand(16)+2
IF mode<>0 THEN
  Plot(64,0)
  DrawTo(192+64,0)
  DrawTo(192+64,191)
  DrawTo (64,191)
  DrawTo (64,0)
FI
RETURN


PROC Instructions()
; ================================
; =          for KoalaPad        =
; = (see listing 2 for joystick)  =
; ================================

Setup(0)
PrintE("              ""Trails!""""")
PutE()
PrintE("      Draw with KoalaPad...")
PutE()
Print("LEFT __\ CLEAR       TOGGLE ")
PrintE("/__ RIGHT")
Print("button / screen      mirror ")
PrintE("\  button")
PutE()
Print("   ANY KEY:")
PrintE("   see INSTRUCTIONS ")
Print("            or ")
PrintE("change PERSISTENCE")
PutE()
PutE()
PutE()
Print("-------------------")
PrintE("-------------------")
IF key=255 THEN
  Print("     Hit any button ")
  Print("to start:")
  DO  UNTIL (PTrig(0)=0 OR PTrig(1)=0
        OR key<>255 OR STrig(0)=0) OD
  key=255
FI
RETURN


PROC Get_pers()

Instructions()    key=255
PutE()
maxpts==/pers_ratio
Print("Persistence is now ")
```

```
PrintCE(maxpts)
PutE()
Print("Enter new ")
Print("persistence (1-10): ")
PrintC(maxpts)  Print("←")
IF maxpts=10 THEN Print("←") FI
maxpts=InputB()
RETURN


PROC Octplot(CARD x0,y0)
CARD x1,y1

x1=191-x0
y1=191-y0
IF Locate(x0+64,y0)=1 THEN
  color=0
ELSE color=1
FI

Plot (x0+64,y0)

IF mirror=1 THEN
  Plot (x0+64,y1)
  Plot (y0+64,x0): Plot (y0+64,x1)
  Plot (x1+64,y0): Plot (x1+64,y1)
  Plot (y1+64,x0): Plot (y1+64,x1)
FI
attract=0
clock=0
IF mirror=0 THEN
  DO  UNTIL ((clock&1)=1) OD
FI
RETURN


PROC Draw()
;==================================
;=          for KoalaPad          =
;=   (see listing 2 for joystick)  =
;==================================

DO

  x0=(Paddle(0)*95/114)-1
  y0=(Paddle(1)*95/114)-1
  IF PTrig(1)=0 THEN
    Setup(24)
    IF mirror=1 THEN mirror=0
    ELSE mirror=1
    FI
    DO UNTIL PTrig(1)<>0 OD
  FI
  IF PTrig(0)=0 THEN
    Setup(24)
  FI
  Octplot(x(n),y(n))
  x(n)=x0  y(n)=y0
  Octplot(x(n),y(n))
  n==+1
  IF n>=maxpts THEN n=0 FI
  UNTIL (key<>255)
OD
RETURN


PROC Main()

Instructions()
DO
  IF (key<>255) THEN
    Get_pers()
    IF maxpts >10 THEN maxpts=10
    ELSEIF maxpts<1 THEN
      Graphics(0)
      RETURN

    FI
    maxpts=maxpts*pers_ratio
  FI
  key=255:  Setup(24)
  Draw()
OD
```

Listing 2.
Action! listing.

```
;      CHECKSUM DATA
;[2A 9D 20 72 ]

PROC Instructions()
; ================================
; =          for joystick        =
; =  (substitute into listing 1)  =
; ================================

Setup(0)
PrintE("              ""Trails!""")
PutE()
Print("          Draw with ")
Print("joystick...")
PutE()
PrintE("   button: CLEAR  and TOGGLE")
PrintE("           screen     mirror")
PutE()
Print("   ANY KEY:")
PrintE("   see INSTRUCTIONS ")
```

```
Print("           or ")
PrintE("change PERSISTENCE")
 PutE()
PutE()
PutE()
Print("------------------------")
PrintE("------------")
IF key=255 THEN
  Print("     Hit any button")
  Print(" to start:")
  DO  UNTIL (PTrig(0)=0 OR PTrig(1)=0
        OR key<>255 OR STrig(0)=0) OD
  key=255
FI
RETURN
```

●

Listing 3.
Action! listing.

```
;      CHECKSUM DATA
;[BA 99 3B ]

PROC Draw()
; ================================
; =          for joystick        =
; =  (substitute into listing 1)  =
; ================================
```

```
INT delx=[0],dely=[0],st

x0=95  y0=95
DO
  st=Stick(0)
  delx=(st&4) RSH 2 - ((st&8) RSH 3)
  dely=((st&1)) -((st&2) RSH 1)
  x0=(x0+2*delx )
  y0=(y0+2*dely )
  IF x0<1 OR x0>190 THEN x0=95  FI
  IF y0<1 OR y0>190 THEN y0=95  FI
  IF STrig(0)=0 THEN
    Setup(24)
    IF mirror=1 THEN mirror=0
    ELSE mirror=1
    FI
    DO UNTIL STrig(1)<>0 OD
  FI
  Octplot(x(n),y(n))
  x(n)=x0  y(n)=y0
  Octplot(x(n),y(n))
  n==+1
  IF n>maxpts THEN n=0 FI
  UNTIL (key<>255)
OD
RETURN
```

●