General Information #

Author: Dave Oblad
Language: ACTION!
Compiler/Interpreter: ACTION!
Published: ANTIC Vol. 4, #4 (08/ 85)

Display Master#

Stretch, squeeze, fold, flop, flip and roll your pictures#

Turn your picture files into silly putty with Display Master. This ACTION! program will stretch, squeeze, fold and otherwise manipulate Graphics 15 pictures, MicoPainter and Micro Ilustrator files. The program requires 48K, a disk drive and the ACTION! cartridge from O.S.S. Antic Disk subscribers will find a run-time version of the program which can be used without the ACTION! cartridge ? LOAD it by going to DOS 2 without BASIC and setting the "L" option on DLMASTER.EXE.

---

*A short while ago, Dave Oblad sent Antic a solution to "The Eight Queens Problem" (April, 1985). As an afterthought be included Display Master. We booted "Eight Queen" liked it, and will publish it in our next issue. But, when we looked at Display Master, it knocked us off our pneumatic computer stools. We think it will do the same for you.?*ANTIC ED.

I was sitting in front of my Atari, which was flipping through a collection of picture files via the Fader program ("Fader II," Antic, May 1985) when I began thinking of those sophisticated special effects used on television, where a picture is squashed, stretched, or goes zooming off into infinity. I wondered if it might be possible to do something similar with my own picture files. So, I set down programming and only a few short weeks later I had Display Master.

THE PROGRAM

Type in the program and save it to disk. Because it uses a good deal of memory; you must compile the program directly from disk. See your ACTION! manual if you're uncertain of this procedure.

When run, Display Master will present you with four options: You can load a new Micro-Painter format picture file, or manipulate a previously loaded picture file, or manipulate a built-in moire pattern, or run a demo program. (We'll discuss Micro Illustrator files later.) For now, choose [D] for the demo program.

A random moire pattern will be drawn on the screen and Display Master will go through its paces. Any time you want to stop, press any keys and you'll return to the option menu.

PICTURE MANIPULATION

Press [P] to load a Micro-Painter file, and you will be asked to place a disk with your pictures in drive 1. These files must be 62 sectors with an extension of .MIC or .PIC. Press [RETURN] when the disk is in the drive and you will see a directory of the files. Type in the device and filename, in the form D:FILENAME.?IC, press [RETURN] and you will arrive at the programming screen.

(If you don't have Micro-Painter files, choose [M] to manipulate a moire screen.)

The programming screen is where You choose your sequence of display manipulation "steps." Display Master offers you 10 manipulation choices and you can program a sequence of up 250 steps. The choices are:

(1) Restore original picture. (2) Invert the picture. (3) Mirror-flip the picture. (4) Shift picture. (5) Double-crush the picture. (6) Fold the picture. (7) Unfold the picture. (8) Roll (by interlacing) the picture 50%. (9) Pause for 2 seconds. (0) Repeat from first step.

Editing this sequence requires the use of 4 keys on the keyboard. Together they allow you to create a series of steps. Each step, when executed, will store the chosen manipulation for that step and move on to the next. The 4 edit keys are:

[-] Minus key to back up a step.
[+] Plus key to move forward a step.
[*] Asterisk key to delete that step.
[=] Equal key to begin execution.

To install a new step simply select an option from 0 to 9 and type that key. The chosen manipulation will be inserted at the current step mumber. A small step window is displayed with the current step pointed to by a "greater than" symbol [>].

For example, type the following sequence:

3939292988670

The pointer should now be pointing to step 14. Press the equal key [=] then [START] and the picture will be loaded into memory then displayed with an ACTION! version of Fader. Your new sequence will then begin. To halt the manipulations, press [START] until the first option menu appears again. From here you can load another picture or return to the editing screen to change the sequence on the currently loaded picture.

MICRO ILLUSTRATOR #

If you don't have Micro-Painter; you can save Micro Illustrator files (the software that comes with KoalaPad, Atari Touch Tablet, etc.) by pressing [INSERT] while in Micro Illustrator. This will save your picture in a 62 sector, uncompacted file called PICTURE.

Unfortunately, the last four bytes of Micro-Painter files contain color register values which will not be saved with this process. To add color to uncompressed Micro Illustrator files, RUN the BASIC program below after placing your color values in the variables in line 10, and you picture filename in line 20.

```
10 K712=66:K708=30:K709=148:K710=196
20 OPEN #1,9,0,"D:PICTURE.PIC"
30 PUT #1,K712:PUT #1,K708:PUT #1,K709:PUT #1,K710
40 CLOSE #1
```

ABOUT THE PROGRAM #

Display Master gets most of its effects by juggling the display list for Graphics 15 (or 7+) mode. Each of the 192 diplay lines are set up with a Load Scan Counter with NNNN Address instruction. The effects are then produced by swapping these byte instructions.

To expedite the swaps, a duplicate set of addresses is maintained in the card array DL(192). Juggling the array and transferring the results to the real display list allows some fast and spectacular displays to be produced. The orginal addresses of the display list are saved in the card array SL(192) for fast restoration of DL(192).

Some manipulations require direct access of the screen memory. For this I defined the memory location of byte array RAM(8000) to overlay the sceen memory, thus giving direct access to the

display data. Byte array ORG(8000) contains the original picture data for fast restoration of the screen.

One other point that should be mentioned involves mirror-flipping the display. Simply reversing the bit order in a displayed byte and swapping byte positions on a line doesn't quite hack it. Remember, the half-nibbles in a given byte define a pixel's color, and reversing the whole byte can really foul up an orginal pixel's color.

The solution was to create a lookup table that uses the original byte value as an index into the table, where the corrected pixel mirror image is stored. This also speed reversing the bit order for a given byte. The table is created near the end of the FIND() procedure and is declared as byte array REV(256).

PROCEDURES USED #

FIND() locates and allocates memory work areas amd makes the reverse table.

MAKE() Makes the display list for Graphics Mode 7.5.

SHOW() Transfers the address list from DL( ) array to the real display list.

RESTORE() Restores the original screen RAM and display list addresses.

FADE() Brings the picture to the screen roughly like the Fader program.

REVERSE() Flips the picture upside down by reversing the display list.

MIRROR() Swaps the pixel positions from each side of the screen.

SHIFT() Wraps the picture around so that the center becomes the edges.

DOUBLE() Removes odd numbered display lines and duplicates to lower half.

UNFOLD() Interlaces even and odd lines in reverse order

FOLD() Interlaces or weaves alternate lines in an ascending sequence.

R0LL() Moves odd numbered lines down and even numbered lines up with rollover at the top and bottowm.

WAIT() A two second dead time to support the pause option.

MOIRE() Generates a semi-random moire pattern In Graphics mode 7.5.

FETCH() Fetches and displays the picture file defined in ARRAY FILE(40).

GETFILE() Prompts user for a file name and displays the disk directory.

DISPLAY() Displays the current step and function assigned to that step.

SETUP() Displays main option menu and supports the step editing.

MAIN() Translates each step into a procedure call.

Dave Oblad is a "non-degreed" Electronic Design Engineer specializing in microprocessor controlled instrumentation. He's been programing in assembly language for eight years.

```
; DISPLAY MASTER
; BY DAVE OBLAD
; (c) 1985, ANTIC PUBLISHING
; (NOTE:COMPILE DIRECTLY FROM DISK!)

BYTE A=$680,B=$681,X=$682,Y=$683
BYTE B0=$684,B1=$685,B2=$686,B3=$687
BYTE C1=$688,C2=$689,C3=$68A,C4=$68B
BYTE D1=$68C,D2=$68D,D3=$68E,D4=$68F
BYTE KEY=$D01F,MASK=$D20A,NOW=$685
CARD SCREEN=$690,DLIST=$692
CARD PNTR=$694,LINE=$696,HOLD=$698
CARD R=$69A,L=$69C
BYTE ARRAY RAM,ORG,REV,STP,FILE
CARD ARRAY DL,SL

PROC FIND()
 GRAPHICS(24)
 DLIST=PEEKC(560)
 SCREEN=PEEKC(DLIST+4)
 RAM=SCREEN
 ORG=RAM-8000
 DL=ORG-400
 SL=DL-400
 REV=SL-300
 STP=REV-300
 FILE=STP-40
 LINE=SCREEN
 FOR X=0 TO 191
  DO
   SL(X)=LINE:LINE==+40
  OD
 DLIST=FILE-1400:DLIST=DLIST&$FC00
 X=0
 DO
  B3=X&$03:B3=B3 LSH 6
  B2=X&$0C:B2=B2 LSH 2
  B1=X&$30:B1=B1 RSH 2
  B0=X&$C0:B0=B0 RSH 6
  B3=B3%B2:B3=B3%B1:B3=B3%B0
  REV(X)=B3
  X==+1:IF X=0 THEN EXIT:FI
 OD
 D1=PEEK(712)
 D2=PEEK(708)
 D3=PEEK(709)
 D4=PEEK(710)
RETURN

PROC MAKE()
 GRAPHICS(24)
 FOR X=0 TO 191
  DO
   DL(X)=SL(X)
  OD
 POKE(DLIST,112)
 POKE(DLIST+1,112)
```

```
 POKE(DLIST+2,112)
 LINE=DLIST+3
 FOR A=0 TO 191
  DO
   POKE(LINE,$4E)
   POKEC(LINE+1,DL(A))
   LINE==+3
  OD
 POKEC(560,DLIST):POKEC(54274,DLIST)
 POKEC(88,SCREEN)
 POKE(LINE,$41)
 POKEC(LINE+1,DLIST)
RETURN

PROC SHOW()
 LINE=DLIST+3
 FOR A=0 TO 191
  DO
   POKEC(LINE+1,DL(A))
   LINE==+3
  OD
RETURN

PROC RESTORE()
 FOR X=0 TO 191
  DO
   DL(X)=SL(X)
  OD
 FOR PNTR=0 TO 7679
  DO
   RAM(PNTR)=ORG(PNTR)
   IF KEY<7 THEN EXIT:FI
  OD
 SHOW()
RETURN

PROC FADE()
 POKE(712,C1)
 POKE(708,C2)
 POKE(709,C3)
 POKE(710,C4)
 FOR X=0 TO 100
  DO
   PNTR=0
   DO
    PNTR==+RAND(60)
    IF PNTR>7679 THEN EXIT:FI
    RAM(PNTR)=ORG(PNTR)&MASK%RAM(PNTR)
    IF KEY<7 THEN EXIT:FI
   OD
  OD
 RESTORE()
RETURN

PROC REVERSE()
 L=0
 FOR Y=0 TO 191
  DO
   R=L+39
```

```
   FOR X=0 TO 19
    DO
     A=RAM(L):B=RAM(R)
     RAM(L)=REV(B)
     RAM(R)=REV(A)
     L==+1:R==-1
    OD
   L==+20
  OD
RETURN

PROC MIRROR()
 FOR X=0 TO 95
  DO
   HOLD=DL(X)
   DL(X)=DL(191-X)
   DL(191-X)=HOLD
  OD
 SHOW()
RETURN

PROC SHIFT()
 PNTR=0
 FOR Y=0 TO 191
  DO
   FOR X=0 TO 19
    DO
     A=RAM(PNTR)
     RAM(PNTR)=RAM(PNTR+20)
     RAM(PNTR+20)=A
     PNTR==+1
    OD
  PNTR==+20
  OD
RETURN

PROC DOUBLE()
 FOR X=0 TO 95
  DO
   DL(X)=DL(X*2)
  OD
 FOR X=0 TO 95
  DO
   DL(X+96)=DL(X)
  OD
 SHOW()
RETURN

PROC UNFOLD()
 X=192
 DO
  X==-1:A=X
  DO
   HOLD=DL(A)
   DL(A)=DL(A-1)
   DL(A-1)=HOLD
   A==-2
   IF A=0 THEN EXIT:FI
   IF A>250 THEN EXIT:FI
```

```
  OD
  SHOW()
  IF X=1 THEN EXIT:FI
  IF KEY<7 THEN EXIT:FI
 OD
RETURN

PROC FOLD()
 X=0
 DO
  A=X
   DO
    HOLD=DL(A)
    DL(A)=DL(A+1)
    DL(A+1)=HOLD
    A==-2
    IF A>250 THEN EXIT:FI
   OD
  SHOW()
  X==+1
  IF X=191 THEN EXIT:FI
  IF KEY<6 THEN EXIT:FI
 OD
RETURN

PROC ROLL()
 FOR X=0 TO 95
  DO
   LINE=DL(191):Y=191
   FOR A=0 TO 94
    DO
     HOLD=DL(Y)
     DL(Y)=DL(Y-2)
     DL(Y-2)=HOLD
     Y==-2
    OD
   HOLD=DL(1)
   DL(1)=DL(0)
   DL(0)=HOLD
   Y=0
   FOR A=0 TO 94
    DO
     HOLD=DL(Y)
     DL(Y)=DL(Y+2)
     DL(Y+2)=HOLD
     Y==+2
    OD
   DL(190)=LINE
   SHOW()
   IF KEY<7 THEN EXIT:FI
  OD
RETURN

PROC WAIT()
 FOR PNTR=0 TO 60000
  DO
   IF KEY<7 THEN EXIT:FI
  OD
RETURN
```

```
PROC MOIRE()
 MAKE()
 POKE(712,D1)
 POKE(708,D2)
 POKE(709,D3)
 POKE(710,D4)
 DO:IF KEY>6 THEN EXIT:FI:OD
 COLOR=1
 Y=RAND(9)+2:X=RAND(9)+2
 Y=Y&$FE:X=X&$FE
 FOR PNTR=0 TO 191
  DO
   PLOT(160,96):DRAWTO(0,PNTR)
   PLOT(159,96):DRAWTO(319,PNTR)
   PNTR==+Y
   IF KEY<7 THEN EXIT:FI
  OD
 FOR PNTR=0 TO 319
  DO
   PLOT(160,96):DRAWTO(PNTR,0)
   PLOT(160,95):DRAWTO(PNTR,191)
   PNTR==+X
   IF KEY<7 THEN EXIT:FI
  OD
 FOR PNTR=0 TO 7679
  DO
   ORG(PNTR)=RAM(PNTR)
   IF KEY<7 THEN EXIT:FI
  OD
RETURN

PROC FETCH()
 MAKE()
 CLOSE(1)
 OPEN(1,FILE,4,0)
 FOR PNTR=0 TO 7679
  DO
   A=GETD(1):ORG(PNTR)=A
  OD
 C1=GETD(1)
 C2=GETD(1)
 C3=GETD(1)
 C4=GETD(1)
 CLOSE(1)
 FADE()
RETURN

PROC GETFILE()
 GRAPHICS(0)
 PRINTE(
"PUT DISK WITH PICTURE FILES IN DRIVE1")
 PRINTE("PUSH ÒÅÔÕÒÎ KEY WHEN DONE!")
 INPUTS(FILE)
 CLOSE(2)
 OPEN(2,"D:*.?IC",6,0)
 DO
  INPUTSD(2,FILE)
  PRINTE(FILE)
```

```
  IF FILE(2)#$20 THEN EXIT:FI
 OD
 CLOSE(2)
 PRINTE("")
 PRINTE("ENTER 'D:FILENAME.EXT'")
 PRINTE("OR JUST HIT ÒÅÔÕÒÎ TO EXIT.")
 PRINT("?"):INPUTS(FILE)
RETURN

PROC DISPLAY()
 POSITION(1,18):PRINT(">")
 POSITION(2,17)
 NOW==-1
 FOR X=1 TO 3
  DO
   PRINT("STEP ")
   PRINTB(NOW)
   PRINT(" IS OPT                  ")
   POSITION(18,16+X)
   IF STP(NOW)>47 AND STP(NOW)<58
      THEN PRINTB(STP(NOW)-48):FI
   IF STP(NOW)=80 OR STP(NOW)=76
      THEN PRINT(FILE):FI
   IF STP(NOW)=77
      THEN PRINT("MOIRE"):FI
   IF STP(NOW)=0
      THEN PRINT("STOP"):FI
   PRINTE("")
   NOW==+1
  OD
 NOW==-2
RETURN

PROC SETUP()
 GRAPHICS(0):POKE(752,1)
 POKE(763,255):POKE(764,255)
 CLOSE(1)
 OPEN(1,"K:",4,0)
 PRINTE
("Display Master, by Dave Oblad")
 PUTE()
 PRINTE ("Antic Magazine, 8/85")
 PUTE() PUTE() PUTE()
 PRINTE
("TO MANIPULATE A ÐICTURE FILE PUSH 'P'")
 PRINTE
("TO MANIPULATE A ÌOADED PICT. PUSH 'L'")
 PRINTE
("TO MANIPULATE A ÍOIRE SCREEN PUSH 'M'")
 PRINTE
("TO SEE A ÄEMO OF THIS PROGRAM PUSH 'D'")
 PRINT
("PUSH 'P' OR 'L' OR 'M' OR 'D' !")
 POKE(702,64)
 A=GETD(1)
 IF A#76 THEN
   IF A=80 THEN STP(0)=A:GETFILE()
    ELSE FILE(1)=0
   FI
```

```
 ELSE STP(0)=A
FI
IF FILE(1)#68 THEN STP(0)=77:FI
IF A#80 AND A#77 AND A#76 THEN
 STP(0)=77
 RETURN
FI
GRAPHICS(0):POKE(752,1)
PRINTE("")
PRINTE("1 = RESTORE DISPLAY")
PRINTE("2 = INVERT DISPLAY")
PRINTE("3 = MIRROR DISPLAY")
PRINTE("4 = SHIFT DISPLAY")
PRINTE("5 = DOUBLE DISPLAY")
PRINTE("6 = FOLD DISPLAY")
PRINTE("7 = UNFOLD DISPLAY")
PRINTE("8 = ROLL DISPLAY 50%")
PRINTE("9 = WAIT 2 SECONDS")
PRINTE("0 = REPEAT FROM STEP 1")
PRINTE("")
PRINTE("- = BACK 1 INSTRUCTION")
PRINTE("+ = SKIP TO NEXT INSTRUCTION")
PRINTE("* = DELETE THIS INSTRUCTION")
PRINTE("= = BEGIN EXECUTION")
NOW=1
DO
 DISPLAY()
 POSITION(2,22)
 PRINT("CHOOSE AN OPTION:")
 A=GETD(1):PUT(A)
 IF A=45 AND NOW>1 THEN NOW==-1:FI
 IF A=43 AND NOW<250 THEN NOW==+1:FI
 IF A>47 AND A<58 THEN
  X=253
  DO
   STP(X)=STP(X-1):X==-1
   IF X=NOW THEN STP(X)=A:EXIT:FI
  OD
  NOW==+1
 FI
 IF A=42 THEN X=NOW
  DO
   STP(X)=STP(X+1):X==+1
   IF X=254 THEN EXIT:FI
  OD
 FI
 IF A=61 THEN EXIT:FI
OD
CLOSE(1)
GRAPHICS(0):POKE(752,1):PRINTE("")
PRINTE
("ÈÏÌÄ DOWN ÓÔÁÒÔ KEY WHEN TONE SOUNDS")
PRINTE
("OR WHEN YOU WANT OPTIONS AGAIN.")
PRINTE
("")
PRINTE
("  PUSH ÓÔÁÒÔ KEY NOW TO BEGIN TASKS!")
DO:IF KEY<7 THEN EXIT:FI:OD
```

```
RETURN

PROC MAIN()
 FIND()
 FOR X=0 TO 254
  DO
   STP(X)=0
  OD
 SCOPY(STP,"88766777666878868740")
 DO
  NOW=1:SETUP()
  IF STP(0)#76 THEN
    IF STP(0)=80 THEN FETCH()
    ELSE MOIRE()
    FI
   ELSE
    MAKE()
    POKE(712,C1)
    POKE(708,C2)
    POKE(709,C3)
    POKE(710,C4)
    RESTORE()
  FI
  NOW=1
  DO
   IF STP(NOW)=48 THEN NOW=1:FI
   IF STP(NOW)=49 THEN RESTORE():FI
   IF STP(NOW)=50 THEN MIRROR():FI
   IF STP(NOW)=51 THEN REVERSE():FI
   IF STP(NOW)=52 THEN SHIFT():FI
   IF STP(NOW)=53 THEN DOUBLE():FI
   IF STP(NOW)=54 THEN FOLD():FI
   IF STP(NOW)=55 THEN UNFOLD():FI
   IF STP(NOW)=56 THEN ROLL():FI
   IF STP(NOW)=57 THEN WAIT():FI
   IF STP(NOW)=0 THEN EXIT:FI
   IF KEY<7 THEN EXIT:FI
   NOW==+1
   POKE(77,1)
  OD
  DO
   A=PEEK(20):B=A RSH 5:B=B&1
   IF B=0 THEN SOUND(0,40,10,8):FI
   IF B=1 THEN SOUND(0,80,10,4):FI
   IF KEY<7 THEN EXIT:FI
  OD
  SNDRST()
 OD
RETURN
```