General Information #

Author: David Plotkin
Language: ACTION!
Compiler/Interpreter: ACTION!
Published: ANTIC Vol. 4, #1 (05/ 85)

AMAZING#

A challenging maze chase game that demonstrates the speed and versatility of the ACTION! language. Requires ACTION! cartridge from Optimized Systems Software. Works on all Atari memory computers with 32K disk or 24K cassette. Antic Disk subscribers will find a "run-time" version on their disk, for playing without the cartidge.

Amazing is a surprisingly imaginative maze chase game written in ACTION! You are a skinny red X named Gork. All you want from life is to wander the city grid, munching up the energy pellets that the programmer thoughtfully left strewn about.

Not surprisingly, three enemies will attempt to stop you with their instantly lethal touch. Luckily, your unique defensive mines can immobilize and vaporize enemies. But of course each opponent is quickly replaced by another.

Release a mine by pressing the joystick button. You can have up to four mines on the board at one time. To retrieve an unused mine, touch it. The mines become available again after destroying an enemy. Naturally, higher levels mean tougher opposition.

HOW IT WORKS #

Type in Listing 1 and SAVE a copy before you compile and RUN it.

Now let's look at some of the game's more interesting ACTION! procedures.

DRAW7 directly manipulates the screen bytes to PLOT a point in the specified color. It's considerably faster than the built-in Atari PLOT function.

FASTDRAW is a high speed technique to put a high resolution picture on the screen. It does direct byte manipulation of the screen with no math involved, so it is considerably faster than even **DRAW7**. The value of each byte that makes up the picture is stored in a byte array, and the width, height, x and y coordinates must be passed to the procedure.

The picture itself is generated using **Drawpic** from Artworx. Drawpic turns the picture you design on the screen into BASIC DATA statements, which can be listed to disk; the format can then be modified to fit into an ACTION! program.

MOVEIT moves the player/missile shape defined by byte array **SHAPE** and player number **WHICH** to the specified position on the screen.

BOARDDRAW draws the initial board. It uses **FASTDRAW** and the byte array BLK to put the squares with letter A on the board.

TESTCOL tests for collisions between the various players by sampling the hardware collision registers. It waits for a whole screen to be drawn, then transfers the contents of the collision registers to temporary locations in RAM. The collision registers are then cleared. Checking for collisions is actually done by looking at the temporary locations.

LLOC performs the same function as LOCATE, but much faster.

GOTBUMPED processes the collisions of the enemy players and a mine. The explosion sounds and flashing of the obliterated player are handled by repeated calls to this procedure. It also removes the enemy player from the board and positions is back in its original corner.

MUNCH detects collisions between your player and the energy pellets. It also keeps the sound going and erases the eaten pellet.

CHANGEDIR decides whether to change the direction of an enemy player. It also checks to see if the player can move in the indicated direction. This procedure is only called when the player is in an intersection.

SMARTS determines whether the enemy players are in an intersection.

OUCH is called if your player is caught by an enemy.

CHASE calls **SMARTS** for each layer, and moves the player if it hasn't been destroyed by a mine.

MOVEMAN reads the joystick and moves your player. It checks to see if you can move in the direction you want. If not, then you continue in the direction you are traveling. Thus, you can push the stick in the desired direction before you get to an intersection and then move in that direction when you hit the intersection.

*Avid ACTION! programmer David Plotkin is a veteran of the **Antic** program submission procedure and, on the side, a chemical engineer for Standard Oil of California.*

```
;    AMAZING
;    BY DAVID PLOTKIN
;    ANTIC MAGAZINE

MODULE

CARD SCRLOC=88,HIMEM=$2E5,
     PM_BASEADR,ADRES,ADRESB,SCORE=[0]

INT  DIRX=[2],DIRY=[0],XDIR,YDIR

INT ARRAY PXDR=[0 0 0 0],
          PYDR=[0 0 0 0]

BYTE T=$DA,VCOUNT=$D40B,
     PMHITCLR=$D01E,DMACTL=$22F,
     GRACTL=$D01D,PMBASE=$D407,
     PRIORITY=$26F,X0,Y0,COUNT=[0],
     LV=[5],FT=[150],CD=[20],
     PCLRM=711,COLR0=708,LOUD=[0],
     COLR1=709,COLR2=710,COLR4=712,
     FATE=53770,CURSH=752,
     TXTROW=656,TXTCOL=657,LVL=[1],
     SND1=$D20F,SND2=$D208

BYTE ARRAY YLOCL(80),
     YLOCH(80),RSH2(160),
     PMHPOS(8)=$D000,
     PX(4)=[0 0 0 0],PY(4)=[0 0 0 0],
     BEGX(4)=[0 52 52 196],
     BEGY(4)=[0 38 166 38],
```

```
     PM_WIDTH(5)=$D008,PLPTR,
     PM_MISMASK(4)=[$FC $F3 $CF $3F],
     PCOLR(4)=704,PMTOPF(8)=$D000,
     PMTOP(8)=$D008,PFCOL(8),PCOL(8)

BYTE ARRAY BM(0)=[$C0 $30 $C $3],
           CM(0)=[$0 $55 $AA $FF],
  CHMP1(0)=[0 0 129 129 66 66 36 36
  24 24 24 24 36 36 66 66 129 129 0 0],
  CHMP2(0)=[0 0 129 129 66 66 60 36
  36 36 36 36 36 60 66 66 129 129 0 0],
  CRT(0)=[0 0 129 129 129 195 90 126
  126 165 165 126 126 90 195 129 129
  129 0 0],
  MSTATUS(0)=[0 0 0 0],ESTAT(4),
  MX(0)=[0 0 0 0],MY(0)=[0 0 0 0],
  BLK(0)=['U'U'U'U'Z'¥'Y'e'Z'¥'Y'e'U
        'U'U'U];WIDTH=2,HEIGHT=8

BYTE ARRAY LINE,DUM
BYTE LOW=LINE,HIGH=LINE+1

PROC DLAY(CARD WAIT)
CARD COUNT
FOR COUNT=0 TO WAIT DO OD RETURN

PROC INIT7()
BYTE LOW1,HIGH1,I  CARD SCREEN=LOW1
GRAPHICS(7) COLR0=44 COLR1=196
COLR2=106 COLR4=0 SCREEN=SCRLOC I=0
WHILE I<80 DO YLOCL(I)=LOW1
YLOCH(I)=HIGH1 SCREEN=SCREEN+40 I=I+1
OD
I=0 WHILE I<160 DO RSH2(I)=I RSH 2
I=I+1
OD
RETURN

INT FUNC HSTICK(BYTE PORT)
BYTE ARRAY PORTS(4)=$278
INT ARRAY VALUE(4)=[0 1 $FFFF 0]
RETURN (VALUE((PORTS(PORT)&$C) RSH 2))

INT FUNC VSTICK(BYTE PORT)
BYTE ARRAY PORTS(4)=$278
INT ARRAY VALUE(4)=[0 1 $FFFF 0]
RETURN (VALUE(PORTS(PORT)&3))

PROC UPDATE()
TXTROW=1 TXTCOL=12 PRINTC(SCORE)
RETURN

PROC UPDATESHIP()
BYTE LOOP5
TXTROW=1
FOR LOOP5=1 TO 5 DO TXTCOL=31+LOOP5
IF LV>=LOOP5 THEN PRINT("o")
   ELSE PRINT(" ")
FI OD RETURN
```

```
PROC DRAW7(BYTE X,Y,CLR)
BYTE X1=$A0,Y1=$A1,CLR1=$A2
LOW=YLOCL(Y1)
HIGH=YLOCH(Y1)
T=RSH2(X1)
LINE(T)=(((BM(X1&3)!$FF)&LINE(T))%
        (BM(X1&3)&CM(CLR1)))
RETURN

PROC FASTDRAW(BYTE ARRAY PICTURE
        BYTE WIDTH,HEIGHT,XX,YY)
BYTE LCTR1,LCTR2  CARD LCTR3
FOR LCTR1=0 TO HEIGHT-1
DO LOW=YLOCL(YY+LCTR1) HIGH=YLOCH(YY+LCTR1)
 LCTR2=XX+WIDTH
 LCTR3=(LCTR1+1)*WIDTH-1
 DO
 LINE(LCTR2)=PICTURE(LCTR3)
 LCTR3==-1 LCTR2==-1
 UNTIL LCTR2=XX
 OD
OD RETURN

PROC PMGRAPHICS()
ZERO(PMHPOS,8)
ZERO(PM_WIDTH,5)
DMACTL=$3E PCOLR(0)=52
PM_BASEADR=(HIMEM-$800)&$F800
PMBASE=PM_BASEADR RSH 8
HIMEM=PM_BASEADR+768
PRIORITY==&$C0%17 GRACTL=3
RETURN

CARD FUNC PMADR(BYTE N)
IF N>=4 THEN N=0 ELSE N==+1 FI
RETURN(PM_BASEADR+768+(N*$100))

PROC PMCLEAR(BYTE N)
CARD CTR
BYTE ARRAY PLAYADR
PLAYADR=PMADR(N)
IF N<4 THEN ZERO(PLAYADR,$100)
 ELSE N==-4
FOR CTR=0 TO $100-1
DO PLAYADR(CTR)==&PM_MISMASK(N) OD
FI
RETURN

PROC WINDOW()
BYTE LOOP5
TXTROW=0 TXTCOL=0 CURSH=1
PRINT
("i-----------------------------------i")
FOR LOOP5=1 TO 2 DO
TXTROW=LOOP5 TXTCOL=0 PRINT("|")
TXTCOL=38 PRINT("|")
OD  TXTROW=3 TXTCOL=0
PRINT
```

```
("!-----------------------------------!")

TXTROW=1 TXTCOL=3 PRINT("SCORE: ")
UPDATE() TXTCOL=20 PRINT("MEN LEFT: ")
UPDATESHIP()
RETURN

PROC MOVEIT(BYTE ARRAY SHAPE BYTE
            WHICH,NUM,XX,YY)
ADRES=PMADR(WHICH)+YY
MOVEBLOCK(ADRES,SHAPE,NUM)
PMHPOS(WHICH)=XX
RETURN

PROC PUTMAN()
BYTE LP
FOR LP=0 TO 3 DO
MSTATUS(LP)=0 ESTAT(LP)=0 OD
X0=120 Y0=102 MOVEIT(CHMP1,0,20,X0,Y0)
FOR LP=1 TO 3 DO
PX(LP)=BEGX(LP) PY(LP)=BEGY(LP)
MOVEIT(CRT,LP,20,PX(LP),PY(LP)) OD
RETURN

PROC BORDER()
BYTE L1,L2
FOR L1=0 TO 159 DO
  FOR L2=0 TO 3 DO
  DRAW7(L1,L2,1) DRAW7(L1,L2+76,1)
OD OD
FOR L1=0 TO 79 DO
  FOR L2=0 TO 3 DO
  DRAW7(L2,L1,1) DRAW7(L2+156,L1,1)
OD OD
RETURN

PROC DOTS()
BYTE L1,L2
FOR L2=8 TO 72 STEP 16 DO
  FOR L1=8 TO 156 STEP 8 DO
DRAW7(L1,L2,3) OD OD
FOR L2=16 TO 72 STEP 16 DO
  FOR L1=8 TO 156 STEP 16 DO
DRAW7(L1,L2,3) OD OD
RETURN

PROC BOARDDRAW()
BYTE L1,L2
BORDER()
FOR L1=2 TO 36 STEP 4 DO
  FOR L2=12 TO 68 STEP 16 DO
  FASTDRAW(BLK,2,8,L1,L2)OD OD
DOTS()
RETURN

PROC TESTCOL()
BYTE LL
FOR LL=0 TO 7 DO
PFCOL(LL)=0 PCOL(LL)=0 OD
```

```
DO UNTIL VCOUNT&128 OD
FOR LL=0 TO 7 DO
PFCOL(LL)=PMTOPF(LL)
PCOL(LL)=PMTOP(LL) OD
PMHITCLR=1
RETURN

BYTE FUNC PMHIT(BYTE N,CNUM)
IF N<4 THEN N==+4 ELSE N==-4 FI
IF CNUM<4 THEN
 RETURN((PCOL(N) RSH CNUM)&1)
 ELSE CNUM==&3
 RETURN((PFCOL(N) RSH CNUM)&1)
FI RETURN(0)

BYTE FUNC LLOC(BYTE XX,YY,CLR)
BYTE X1=$A0,Y1=$A1,CLR1=$A2,L1,L2
LOW=YLOCL(Y1) HIGH=YLOCH(Y1)
T=RSH2(X1) L1=X1&3
L2=LINE(T)&BM(L1)
IF (L2&CM(CLR1))=(BM(L1)&CM(CLR1))THEN
RETURN(1) FI;SOMETHING THERE
RETURN(0)

BYTE FUNC LKAHD(INT XD,YD BYTE XX,YY)
BYTE XA,YA,XB,YB,RS1,RS2
XA=XX-48 YA=(YY-32) RSH 1
IF XD>0 THEN XA==+7+XD XB=XA
 YA==+1 YB=YA+7
 ELSEIF XD<0 THEN XA==+XD XB=XA
 YA==+1 YB=YA+7
 ELSEIF YD>0 THEN XB=XA+7
 YA==+9 YB=YA
 ELSEIF YD<0 THEN XB=XA+7 YB=YA
 ELSE RETURN(0)
FI RS1=LLOC(XA,YA,1) RS2=LLOC(XB,YB,1)
IF RS1+RS2=0 THEN RETURN(1)
   ELSE RETURN(0);OK
FI RETURN(0);BLOCKED

PROC NEWLEVEL()
BYTE LL
SNDRST() SCORE==+COUNT*LVL
UPDATE() COUNT=0 LVL==+1
FOR LL=0 TO 7 DO PMCLEAR(LL) OD
DOTS() PUTMAN()
DIRX=0 DIRY=0
IF LVL<11 THEN FT==-10 CD==+10 FI
RETURN

PROC MSLDROP(INT XD,YD)
BYTE TRIG=644,XA,YA,LP,MASK,LD=[0],TT=[0]
IF LD>1 THEN LD==-2 FI
SOUND(1,LD LSH 3,10,LD)
IF TRIG=1 THEN TT=0 FI
IF TRIG=1 OR (XD=0 AND YD=0) OR TT=1
 THEN RETURN FI
FOR LP=0 TO 3 DO
IF MSTATUS(LP)=0 THEN MSTATUS(LP)=1
```

```
 IF XD>0 THEN XA=X0 YA=Y0+9
   ELSEIF XD<0 THEN XA=X0+7 YA=Y0+9
   ELSEIF YD>0 THEN XA=X0+4 YA=Y0
   ELSE XA=X0+4 YA=Y0+18
 FI MASK=PM_MISMASK(LP)!$FF LD=12 TT=1
 MY(LP)=YA MX(LP)=XA
 PLPTR(MY(LP))==%MASK
 PLPTR(MY(LP)+1)==%MASK
 PMHPOS(LP+4)=MX(LP) EXIT
FI OD RETURN

PROC MSLGET()
BYTE LP,LD1=[0]
IF LD1>1 THEN LD1==-2 FI
SOUND(2,LD1 LSH 4,10,LD1)
FOR LP=0 TO 3 DO
IF PMHIT(LP+4,0)=1 THEN
MSTATUS(LP)=0 LD1=12
PLPTR(MY(LP))==&PM_MISMASK(LP)
PLPTR(MY(LP)+1)==&PM_MISMASK(LP)
PMHPOS(LP+4)=0 EXIT FI OD RETURN

PROC GOTBUMPED()
BYTE LQ,LD2=[0],LQ1
IF LD2>0 THEN LD2==-1 FI
SOUND(3,LD2 LSH 3,8,LD2)
FOR LQ=0 TO 3 DO FOR LQ1=1 TO 3 DO
IF PMHIT(LQ+4,LQ1)=1 THEN
 LD2=14 ESTAT(LQ1)=1 MSTATUS(LQ)=0
 PLPTR(MY(LQ))==&PM_MISMASK(LQ)
 PLPTR(MY(LQ)+1)==&PM_MISMASK(LQ)
 PMHPOS(LQ+4)=0
FI OD OD
FOR LQ=1 TO 3 DO
IF ESTAT(LQ)>0 THEN ESTAT(LQ)==+1
 PCOLR(LQ)=FATE
FI
IF ESTAT(LQ)=FT THEN ESTAT(LQ)=0
 PMCLEAR(LQ)
 PCOLR(LQ)=(RAND(15) LSH 4)+6
 PX(LQ)=BEGX(LQ) PY(LQ)=BEGY(LQ)
 MOVEIT(CRT,LQ,20,PX(LQ),PY(LQ))
FI OD RETURN

PROC MUNCH()
BYTE TIME=20,X1,Y1
IF LOUD>1 THEN LOUD==-2 FI
SOUND(0,8,LOUD LSH 3,LOUD)
IF PMHIT(0,10)=0 THEN DLAY(1) RETURN FI
LOUD=12 X1=X0-48 Y1=(Y0-32) RSH 1
DRAW7(X1+3,Y1+4,0) DRAW7(X1+3,Y1+5,0)
DRAW7(X1+4,Y1+4,0) DRAW7(X1+4,Y1+5,0)
COUNT==+1
IF COUNT=135 THEN NEWLEVEL() FI
RETURN

PROC CHANGEDIR(BYTE WH)
BYTE F,LP
IF FATE<CD THEN F=RAND(4)
```

```
 IF F=0 THEN PXDR(WH)=2 PYDR(WH)=0
  ELSEIF F=1 THEN PXDR(WH)=-2
  PYDR(WH)=0 ELSEIF F=2 THEN
  PXDR(WH)=0 PYDR(WH)=2 ELSE
  PXDR(WH)=0 PYDR(WH)=-2
 FI
FI
IF LKAHD(PXDR(WH),PYDR(WH),PX(WH),
 PY(WH))=0 THEN PXDR(WH)==-PXDR(WH)
 PYDR(WH)==-PYDR(WH)
FI RETURN

PROC SMARTS(BYTE WHICH)
BYTE X,Y
X=PX(WHICH) Y=PY(WHICH)
IF (X=52 OR X=68 OR X=84 OR X=100
 OR X=116 OR X=132 OR X=148 OR X=164
 OR X=180 OR X=196) AND (Y=38 OR Y=70
 OR Y=102 OR Y=134 OR Y=166) THEN
 CHANGEDIR(WHICH)
FI RETURN

PROC ENDGAME()
BYTE TRIG=644,ST=755,TIME=20
SCORE==+COUNT*LVL PMHITCLR=0 UPDATE()
COUNT=0 LVL=1 TXTROW=2 TXTCOL=8
PRINT("GAME OVER  ÐÒÅÓÓ ÆÉÒÅ")
DO ST=(TIME RSH 4)&1 UNTIL TRIG=0 OD
LV=5 UPDATESHIP()
SCORE=0 TXTROW=1 TXTCOL=12
PRINT("     ") TXTROW=2 TXTCOL=8
PRINT("                    ")
UPDATE() DOTS() PUTMAN() FT=150 CD=20
XDIR=0 YDIR=0 DIRX=0 DIRY=0 ST=0
RETURN

PROC OUCH()
BYTE LC,LD
IF PCOL(4)=0 THEN RETURN FI
LC=Y0+10  LD=Y0+10
DO LD==+2 IF LD>200 THEN LD=200 FI
LC==-2 IF LC<30 THEN LC=30 FI
IF (LC=30 AND LD=200) THEN EXIT FI
SOUND(0,LC,8,8) SOUND(1,LD,8,8)
DUM(LC)=FATE DUM(LD)=FATE
DLAY(250) DLAY(250) DLAY(250)
OD SNDRST()
FOR LC=0 TO 7 DO PMCLEAR(LC) OD
LV==-1 UPDATESHIP()
IF LV=0 THEN ENDGAME() ELSE PUTMAN()
PMHITCLR=0  FI RETURN

PROC CHASE()
BYTE LP
FOR LP=1 TO 3 DO SMARTS(LP)
PX(LP)==+PXDR(LP) PY(LP)==+PYDR(LP)
IF ESTAT(LP)=0 THEN
MOVEIT(CRT,LP,20,PX(LP),PY(LP)) FI
OD RETURN
```

```
PROC MOVEMAN()
BYTE STCK=632,TIME=20
XDIR=HSTICK(0) LSH 1
YDIR=VSTICK(0) LSH 1
IF XDIR<>0 AND YDIR<>0 THEN YDIR=0 FI
IF STCK=15 THEN XDIR=DIRX YDIR=DIRY FI
IF LKAHD(XDIR,YDIR,X0,Y0)=1 THEN
 X0==+XDIR
 Y0==+YDIR DIRX=XDIR DIRY=YDIR ELSEIF
 LKAHD(DIRX,DIRY,X0,Y0)=1 THEN
 X0==+DIRX Y0==+DIRY
 ELSE DIRX=0 DIRY=0
FI MOVEIT(CHMP1,0,20,X0,Y0)
RETURN

PROC MAIN()
BYTE XX,COUNT,TIMER=20,ATRACT=$4D
SND1=3 SND2=0
INIT7() PMGRAPHICS() PCLRM=50
PLPTR=PMADR(4) DUM=PMADR(0)
FOR XX=0 TO 7 DO PMCLEAR(XX) OD
FOR XX=1 TO 3 DO
PCOLR(XX)=(RAND(15) LSH 4)+6    OD
WINDOW() BOARDDRAW() PUTMAN() ENDGAME()
DO TESTCOL() MUNCH() MOVEMAN() OUCH()
MSLGET() CHASE() MSLDROP(DIRX,DIRY)
ATRACT=0 GOTBUMPED() OD
RETURN
```

Game AMAZING in ACTION/amazing.djvu
Game AMAZING in ACTION/amazing.PDF