At 25, Mike Stortz is the P.D. Librarian for GRASP, the Richmond, Virginia Atari users group. Seemingly unable to find work in the programming field, he is working on about thirty projects at once, including a graphic arcade/adventure game.

by Mike Stortz

1996: Kara Hyke leaves show business after losing out in a disputed Oscar award.
1997: Kara Hyke enrolls at M.I.T.
2010: Kara Hyke discovers semi-matter.
2016: Hyke-Grey effect discovered.
2030: Hyke and Grey found Arcadia on Proxima III.
2065: Hammer scout ship attacks Arcadia and is driven off after extensive damage to the city. Work on the shield begins.

Today: Hammer fleet attacks Arcadia...

Deep in caverns under Arcadia, fuel cannisters of semi-matter ore to power the city's defensive shield have been cached away against the day when the aliens would return. Composed of mixed-charge matter, hyke (as it is called) is easily persuaded to annihilate itself in the manner of matter-antimatter reactions, but is more easily stored.

Now that the attack has begun, brave volunteer retrievers must don helipacks, fly down into the caves and bring up the ore, so that Arcadia may not fall.

(continued on page 27)
R.O.T.O. is a game of coordination, reflexes, timing and a hint of strategy that should keep you going for a while before Arcadia can claim victory.

Plug in your Action! cartridge and type in Listing 1. Action! is forgiving about case, spacing and line divisions, so you need not slavishly follow the format of the listing (which is compressed somewhat for purposes of publication).

SAVE it before attempting to RUN it! If you try to run R.O.T.O. from memory, the source code will be overwritten and ruined, causing much gnashing of teeth. Because R.O.T.O. is so large, it must be compiled off of disk or cassette.

For disk.

After typing R.O.T.O. in, save it with the command CTRL-W and type in the filename "ROTO.ACT" then RETURN. SHIFT-CLEAR the editor, enter the monitor with the CTRL-M command, enter C "ROTO.ACT" and RETURN. This will compile R.O.T.O. into machine language. Now, save the compiled version by typing W "ROTO.AML" and RETURN. AML stands for Action! machine language. To run R.O.T.O. now, just type R and RETURN. In the future, simply type CTRL-M to enter the monitor, D and RETURN to go to DOS, and then binary loading the file ROTO.AML from DOS by using the L command in DOS 2 or DOS 3—or type LOAD ROTO.AML if you have DOS-XL.

For cassette.

Type in R.O.T.O. and save it with the CTRL-W command. Do not use the "Screen Off?" option; it will upset the tape timing. Rewind your tape, press PLAY and RECORD, give the filename 'C:' and RETURN.

Go have lunch while the source code is being saved (about fifteen minutes). Come back, clear memory with the SHIFT CLEAR, and enter the monitor with the CTRL-M command.

Rewind the tape containing the R.O.T.O. source code, press PLAY, type in R "C:" and RETURN. Have some more lunch. The computer is rereading the source code, compiling it as it does. When it's finished compiling, the game will automatically start.

In the future, you may play R.O.T.O. by inserting the tape containing the source code into the recorder and typing the R "C:" command.

Playing R.O.T.O.

After beginning, you should see the R.O.T.O. logo and your man flying about it, while an explanatory message scrolls beneath. You may begin by pressing the START button or the fire button on joystick 1. You will see a portion of a cavernous network and four green blocks with Hs on them. This is the fuel intake.

Cannisters of hyke are scattered about the caves. They look like the fuel intake, except that they're glowing. Pick up these cannisters by touching them, then return to the intake and touch it. This advances your score and charges the shield in accordance with however many cannisters you've deposited.

Each cannister is worth fifty points. Returning ore also refuels your helipack. Picking up more than ten cannisters before depositing them will cause their magnetic fields to interact with explosive results.

Your man moves up, down, left, right and diagonally in all directions, although he moves downwards faster than up. Moving against the screen's border will scroll more caverns into view, although you'll automatically stop at the far ends of the caves.

Don't run into a wall, or you'll lose a helipack. Watch your fuel, too. Running out will produce the expected effect.
Your retriever is also equipped with a molecular debonding device to facilitate going through rock. Press the fire button to let off a shot. The debonder will vaporize any chunk of rock you fire on, but you will lose one point for every piece of the cave you eliminate (because you're reducing their structural integrity).

Unfortunately, the debonder will also affect a fuel cannister. Rupturing the magnetic bottle containing the hyke will produce a large explosion and prevent anyone else from retrieving fuel.

Also, attacks from the alien fleet will shake the caves periodically, causing rocks to break loose from the ceiling. Shooting rocks is worth one point each (for cleaning up). Don’t run into them, and be careful that the tremors don’t send you into a wall.

Pressing any key while a game is in progress will pause it. Press another key to resume play. If you want to begin again, press START.

If you play well and retrieve enough cannisters to top 1000 points, the shield will have stayed up long enough for reinforcements to arrive—and the city will be saved.

On the other hand, if you wreck while hauling cannisters, you could deplete the ore supply so that victory is impossible.

Remember which portions of the caverns you’ve mined out, and definitely recall the way back to the fuel intake. The cave network is generated randomly each game, so expect variety.

**R.O.T.O.** may end in five ways:

1. Most frequently, you run out of helipacks (while there are many volunteers, there are only three of the sophisticated flying apparatuses).
2. The shield is battered down. This happens when you don’t retrieve enough ore.
3. A fuel cannister is shot.
4. You carry more than ten cannisters at once.
5. The least common... Arcadia holds out, and the aliens are defeated—this time!

**That’s it!**

Action! deserves a word of praise here. **R.O.T.O.** was designed half in advance and half as I thought of another feature to put in. The excellent editor made even major reshuffling and splitting of routines easy.

I shudder to think what I would have gone through using a conventional assembler. Even when the source code became too large to co-reside with the object code, I could compile off of my Axlon RAMDisk with little loss of development time.

The author would appreciate any letters of business offers, extravagant praise, or, failing that, constructive comment. Have fun and save Arcadia!

---

**Listing 1.**

**Action! listing.**

```
; R.O.T.O. by Mike Stortz
SET $808E=$4080
SET $8491=$4080

DEFINE bytes="64", lines="88", rock="194", pwb_page="128", ch_page="128", ch_adr="3276",
dl_page="136", dl_adr="34816",
MISC_page="137", MISC_adr="35872",
SC_page="140", SC_adr="35840"

BYTE
rtclock=$14, attract=$49, Imargin=$52,
rowcrs=$54, dindex=$57, smctc=$22F,
gprior=$26F, crsinc=$2F8,
ch=$2FC, gpracti=$DO1D,
hitcr=$DO1E, consol=$DO1F,
audcti=$DO20, skystat=$DO28F,
pwbasc=$DO40, wsync=$DO40,
vcount=$DO48, nmien=$DO48E,
chbas=$2F4, chbase=$DO49,
hsrcl=$DO4A, vsrcl=$DO4B,
pscr1=$200, pscr1=$2C1,
pscr2=$2C2, pscr3=$2C3,
colp0=$DO12, colpm=$DO13,
colm2=$DO14, colm3=$DO15,
colr0=$2C4, colr1=$2C5,
colr2=$2C6, colr3=$2C7,
colr4=$2C8,
colpF=$DO16, colpf=$DO17,
colp2=$DO18, colp3=$DO19,
colpF4=$DO18

BYTE ARRAY hpos=$D800, wpx=$D800,
hposn=$D804, wpnx=$D804

CARD colrs=$55, savmsc=$56,
udmlst=$200, dsmllst=$230,
txtmsc=$294

BYTE i,j,k,l,cx,cy,x,y,xs,xsm,ys,joy,
phs,mc,mc,mc,flag,bak,fore,
fuel,packs,enable,whine,
carried,end,cans,fallc,shake,
shakec
```

---

PAGE 26 / JUNE 1985

ANALOG COMPUTING
PROC CheckShake()
    IF Rand(0)=255 AND
        Rand(3)=0 AND
        shake=0 THEN
        shake=Rand(10)+10
        shield=Rand(20)
        IF shield=0 THEN shield=0 FI
        IF shake=0 THEN end=2 FI
        Plot(159,7) DrawTo(shield,7)
    FI
ELSE
    shake=60
    IF shake=0 THEN
        shake=1
        Rand(10)
        IF fall(j)=0 THEN
            a=table(cy)+cxRand(20)
            IF screen(a)=0 THEN
                fall(j)=a id(j)=rock
                screen(a)=rock
            FI
        FI
        sound(2,255-shake,2,6)
        xsm=Rand(5)
    ELSE
        xsm=0
        sound(2,0,0,0)
    FI
RETURN

PROC DoScore()
    IF rtclock=0 OR rtclock=120 AND
        fuel=0 THEN
        DoScore() fuel=-1
    FI
RETURN

PROC EndGame()
    Zero(Misc_adr+80,80) ZeroOut()
    dindex=2 rowcrs=4 colcrs=0
    IF end=1 THEN
        PrintDEG(" NO PACKS LEFT")
    ELSEIF end=2 THEN
        PrintDEG(" SHIELD DEPLETED")
    ELSEIF end=3 THEN
        PrintDEG(" CANNISTER RUPTURED")
    ELSEIF end=4 THEN
        PrintDEG(" TOO MANY CANNISTERS")
    ELSEIF end=5 THEN
        PrintDEG(" ARCADIA THANKS YOU")
    FI
    PutDEG(6) PrintDEG(" game over")
FOR a=1 TO 400 DO
    Sound(0,a RSH 1.8,6)
    DO UNTIL vcount=128 OD
    FOR i=0 TO 60 DO
        colp80=vcount rtcl
        mspin=0
    OD
    OD
RETURN

PROC GetDir()
PROC Scroll()

IF (joy&8) <> 0 THEN x5 := +1 x3 := +1
  IF x5 = 0 THEN
    IF cx = 0 THEN x5 := -1
    ELSE cx := -1 x5 := 0
  FI
  ELSEIF (joy&8) <> 0 THEN x5 := -1 x3 := -1
  IF x5 = 255 THEN
    IF cx = 44 THEN x5 := +1
    ELSE cx := +1 x5 := 7
  FI
  FI
  IF (joy&2) <> 0 THEN y5 := +1 y3 := -2
  IF y5 = 0 THEN
    IF cy = 68 THEN y5 := -1
    ELSE cy := +1 y5 := 0
  FI
  ELSEIF (joy&1) <> 0 THEN y5 := -1 y3 := +1
  IF y5 = 255 THEN
    IF cy = 0 THEN y5 := +1
    ELSE cy := -1 y5 := 7
  FI
  FI
  DO UNTIL wxcount = 128 OD
  ary := 8a a := screen + table(cy) + cx j := 12
  FOR i := 0 TO 17 DO
    list (j + i) := ary (i)
    list (j + i) := ary (1)
    j := +3
    a := + bytes
  OD
  RETURN

PROC MoveMan()

Zero (pmb + y, 26) Zero (pmb + 256 + y, 26)
  x := + dx y := + dy
  hposp (0) := x hposp (1) := x
  IF x <> 0 THEN face := 0
  ELSEIF x <> 0 THEN face := 1
  FI
  ary := pmb + y phase := RSH 2
  temp := manadr (face)
  MoveBlock (a, man0 + temp, 21)
  MoveBlock (a + 256, man1 + temp, 21)
  ary := pmb + y + 1
  a := + phase := RSH 2
  i := rotor (rotoradr (face) + phase)
  Poke (i, 1) Poke (i + 256, i)
  Sound (0, phase := LSH 2 - (yd := LSH 3) + 8, 2)
  RETURN

PROC GoBoom()

SndRst()
  Zero (missile, 256) mx (0) := 0 mx (1) := 0
  Wait (30)

ary := pmb + y
FOR i := 0 TO 170 DO
  FOR j := 1 TO 20 DO
    colpmb := 64 * Rand (8) LSH 1
colpmw := 64 * Rand (8) LSH 1
wsync := 0
  OD
k := Rand (24) ary (k) := & Rand (0)
k := Rand (24) ary (k + 256) := & Rand (0)
sound (i, i, i, 4, 6)
Wait (1)
  OD
Zero (pmb, 512)
SndRst () pcolr0 := 152 pcolri1 := 118
Wait (20) enable := 0
FOR i := 0 TO 14 step 2 DO
  fore := 46 - i
  Wait (5)
  OD
fore := 0 Wait (60) hitclr := 0
carried := 0 whine := 0 shake := 0 face := 0
packs := -1
  IF packs = 0 THEN end := 1 FI
  FOR i := 0 TO 19 DO
    screen (fall (i)) := 0
    fall (i) := 0
  OD
  fuel := 50 color := 1
  Plot (0, 5) DrawTo (fuel, 5)
color := 0
  x := 84 y := 110 cx := 8 cy := 0 xs := 7 ys := 0
  DoScore () Scroll () MoveMan ()
  fore := 36 enable := 1
RETURN

PROC GetCan()

i := -35 j := y := 5
i := RSH 3 j := RSH 3
a := table (j + cy) + i + cx
  IF screen (a) = 157 THEN
    screen (a) := 0
    carried := -1
    IF carried = 0 THEN end := 4 FI
    whine := 200 hitclr := 0
  FI
RETURN

PROC Falling (CARD bb)

j := screen (bb - 64)
  IF j = 159 OR j = rock THEN
    FOR k := 10 TO 19 DO
      IF fall (k) = 0 THEN
        fall (k) := bb - 64 id (k) := j EXIT
      FI
    OD
  Falling (bb - 64)
FI
RETURN

PROC ZapIt (BYTE zz)

affect := 0
l := mx (zz) + 2
j := mx (zz) - 31 - RSH 2 - xs
k := wy (zz) - 72 + y
j := RSH 3 k := RSH 3
missile (wy (zz)) := & 255 - wdata (zz)
mx (zz) := 0

ANALOG COMPUTING JUNE 1985 / PAGE 31
a = table(cy+j)+cx+j
IF screen(a)=159 THEN end=3 FI
IF screen(a)=rock THEN score=+2 FI
bak=70 fore=-12
FOR j=0 TO 10 DO
    screen(a)=65
    FOR k=1 TO 100 DO OD
    screen(a)=0
    FOR k=1 TO 100 DO OD
    Sound(1,280,2,15,1)
    OD
bak=80 fore=36 screen(a)=0
Sound(1,0,0,0)
hitcr=0 score=-1 DoScore()
Falling(a)
RETURN

PROC MoveRocks()
FOR i=0 TO 19 DO
    temp=fall[i]
    IF temp>0 THEN
        IF screen(temp)=0 THEN
            temp=0
            ELSE
                a=temp+64
                IF screen(a)<0 THEN
                    temp=0
                IF id(i)=159 THEN end=3 FI
                ELSE
                    screen(temp)=0
                    screen(a)=id(i)
                temp=+64
            FI
            FI
        fall(i)=temp
        OD
    OD
RETURN

PROC CheckRocks()
fallc=-1
IF fallc<0 THEN
    fallc=20 MoveRocks()
    FI
RETURN

PROC DrawWall(CARD st BYTE cc,in,len)
BYTE ii,jj
CARD tt
screen(st)=1
tt=st+in
FOR ii=1 TO len-2 DO
    jj=Rand(2)
    screen(tt)=cc+jj
    tt=tt+in
    OD
screen(tt)=1
RETURN

PROC DrawCaves()
sdwx=0 sdyst=dlist
Zero(scx,adr,5120)
Zero(scvn,adr,512)
FOR i=0 TO 11 DO
dlist(i)=ddata(i)
OD
a=screen j=12
FOR i=0 TO 17 DO
dlist(j)=64+32+16+6
dlist(j+1)=a&$FF
dlist(j+2)=a RSH 8
j++;a++bytes
OD
dlist(j-3)=128+64+16+6
FOR i=0 TO 7 DO
dlist(j+1)=ddata2(i)
OD
txtmsg=mcx,adr
FOR i=0 TO 7 DO
    Poke(mcx,adr+17+i,79+i)
Poke(mcx,adr+57+i,87+i)
OD
rowcrs=2  colcrs=8
PrintE("fuel packs:")
colors=8
PrintE("shield score:")
PrintE(" by mike stortz")
dindex=6  color=1
Plot(0,5) DrawTo(fuel,5)
Plot(0,7) DrawTo(shield,7)
a=0
FOR i=0 TO 7 DO
    FOR j=0 TO 15 DO
        k=Rand(32)
        IF (k&16)<0 THEN k=%4 FI
        IF (k&1)<>0 THEN
dataWall(a,3,1,4) FI
        IF (k&2)<0 THEN
dataWall(a+1,5,bytes,10) FI
        IF (k&4)<0 THEN
dataWall(a+4,7,1,4)
            IF Rand(5)=0 THEN
                screen(a+5)=rock
            FI
            IF (k&8)<0 THEN
dataWall(a,9,bytes,10) FI
            IF (k&16)<0 AND
                j=0 AND j<15 THEN
                screen(a+5)=159 FI
                a+=4
            FI
            a+=576
        OD
        screen(19)=95
        screen(79)=95
        screen(133)=95
        screen(134)=95
        PMSet()
    RETURN
    PROC Title()
    BYTE t
    Graphics(21)
    PMSet() ZeroOut()
    Zero(missile,1280)
    Zero(misc_adr,3800)
    screen=savmsc dist=sds1st
colr=158
color=146
color2=48
color3=68
color4=64
k=0
    FOR i=6 TO 13 DO
        FOR j=8 TO 15 DO
            screen(j+20+i)=logo(k)
            k+=1
            OD
        OD
        FOR i=6 TO 13 DO
            FOR j=16 TO 23 DO
                screen(j+20+i)=logo(k)
                k+=1
            OD
        OD
dist(31)=32
dist(32)=64+32+6
dist(33)=0
dist(34)=3
b=misc_adr
FOR i=35 TO 44 DO
    dist(i)=32+6
    OD
dist(44)=6
FOR i=45 TO 52 DO
    dist(i)=0
    OD

    dindex=8  lmargin=1
dist(10)=6
savmsc=+400
PrintD(6," DRAW")
PrintD(6,scree)
colors=10
PrintD(6,"DASH")
PrintD(6,high)
savmsc=misc_adr+300
rowcrs=0  colcrs=1

PrintDE(6," reserve ore")
PrintDE(6," transport (operation")
PrintDE(6," THE CITY OF")
PrintDE(6," ARCADIA IS UNDER")
PrintDE(6," ATTACK, YOUR JOB")
PrintDE(6," IS TO RECOVER FUEL")
PrintDE(6," CANNISTERS OF HYMN")
PrintDE(6," AND RETURN THEM")
PrintDE(6," THE UPPER LEFT END")
PrintDE(6," OF THE CAVERNS, IF")
savmsc=+400
rowcrs=0  colcrs=1
PrintDE(6,"YOUR SCORE EXCEEDS")
PrintDE(6,"1000, ARCADIA HAS")
PrintDE(6,"HELD OUT LONG")
PrintDE(6,"ENOUGH FOR HELP TO")
PrintDE(6,"ARRIVE, DON'T")
PrintDE(6,"SHOOT A CANNISTER")
PrintDE(6,"CARRY MORE THAN")
PrintDE(6,"AND DO NOT RUN INTO")
PrintDE(6,"WALL, GOOD LUCK")
savmsc=+400
rowcrs=0  colcrs=1
PrintDE(6," PRESS START")
PrintDE(6," TO PLAY")
x=86  y=58  xd=-1  yd=0  ys=0  xs=0
phase=0  t=0
DO
    IF yd<1 THEN
        yd=0  xd=2  gprior=36
    ELSEIF yd<1 THEN
        yd=0  xd=2  gprior=33
    ELSEIF xd<2 THEN
        xd=0  yd=1
    ELSEXD=8  yd=1
    FI
    FOR t=0 TO 39 DO
        IF xd>2 AND x=158 THEN
            gprior=36
        FI
        IF xd>2 AND x=116 THEN
gprior=33
        FI
        MoveMan()
        DoPhase()
        1:=1
        IF l=2 THEN
            l=0  y=:=1
        FI
        IF y=8 THEN
PRINTE(6," DRAW")
PRINTE(6,"DASH")
PRINTE(6,high)
R.O.T.O. continued

ys=0
b=120
IF b=misc_adr+1320 THEN
b=misc_adr FI
DO UNTIL ycount=128 OD
dist[i3]=peek(0b)
dist[i3]=peek(0b+1)
FI
Wait(2)
IF consol=6 OR Strig(0)=0 THEN
EXIT
FI
DO
UNTIL consol=6 OR Strig(0)=0 OD
ch=255 SndRst()
RETURN

PROC Init()

skstat=3 audctl=0 high=0 score=0

MoveBlock(cb_adi,57344,1024)
MoveBlock(cb_adi,1210,cs7,80)
MoveBlock(cb_adi,1264,cs7,80,288)
MoveBlock(cb_adi,1264,cs7,288,120)
MoveBlock(cb_adi,1264,logo,128)
MoveBlock(cb_adi,760,can,0)

pmb=pmb_page*256+1024
Missile=pmb--256

a=0
FOR i=0 TO 79 DO
	table[i]=a
	a+=bytes
OD

setuvt(7,ublank R5H 8,ublank&$FF)
vsdlst=di nwiem=192
RETURN

PROC LoopInit()

colr0=68 colr1=40 colr4=64
screen=5c_adi dist=di_adi
save=mc=misc_adi
PmSet() ZeroOut()
Phase=0 face=0 MC=1 flag=0
Cx=0 cy=0 x=7 ys=0 x=84 y=110
bak=0 fore=36 enable=1 fallc=1

packs=3 score=0 fuel=50 shield=50
whine=0 carried=0 end=0
Shake=0 Shake=0 XSN=0
FOR i=0 TO 19 DO
	id[i]=0 fallc[i]=0
OD
wx(0)=0 wx(1)=0
RETURN

PROC GameLoop()

vsdlst=di nwiem=192
GetDir()
IF fuel=0 THEN
	yd=2 joy=+$2
ELSE

dPhase()
FI

MoveMan()
IF x<70 OR x>176 OR

g<90 OR g>172 THEN

scrol1()
ELSE

Wait(1)
FI

CheckFuel()
CheckShake()
CheckRocks()
StartMiss() MoveMiss() Bump()
IF whine<>0 THEN

whine=-1
sound(3,whine,10,4)
ELSE

sound(3,0,0,0)
FI
IF ch<255 THEN

ch=255 5ndRst()
DO UNTIL ch<255 OR

consol<>7 OR

Strig(0)=0 OD
ch=255
FI
RETURN

PROC Game()

Init()
DO
Title()
Graphics(0) crssinh=1
LoopInit()
DrawCavest()
DoScore()
color=0
DO
GameLoop()
UNTIL consol<>7 OR end<>0 OD
IF end<>0 THEN EndGame()
FI
IF score<>0 AND score>high THEN

high=score
FI
OD
Setuvt(7,5E4,562)
ZeroOut() Graphics(0)
RETURN