Sneak Attack

by David Plotkin

You knew it had been too quiet. Nothing had shown on the scanner for the whole watch. That in and of itself wasn't unusual, but intelligence had reported increased enemy activity. It seemed that a major move to capture and destroy the gunbases that protected the interior was being planned.

Further, the enemy had developed a new type of intelligent robot, which could stand the shock of being parachuted to Earth and, once there, could team up with other robots to destroy the gunbases. Intelligence reports indicated that each robot could carry one-quarter of the explosives necessary to pierce the armor of the gunbase you manned.

The anticipated plan is that the enemy choppers will drop robots, which, if they land successfully, will wait until three more robots have also landed, then team up to destroy your base. Since radio silence must be maintained, the robots only "know" about other robots in their direct line of sight, so four robots must land successfully on one side of your base.

These robots are not invulnerable. However, if one parachuting robot lands on another, the one underneath will be crushed and immobilized.

Your gunbase is a pretty awesome weapon. The gun is mounted on a concrete pedestal and is aimed by your joystick. The missiles unleashed by your fire button are steerable—they will travel left and right.
if you press the joystick control in the appropriate direction, and rise toward the top of your scanner screen if you center the stick.

The missiles are powerful, capable of obliterating the enemy's helicopters, as well as the robots. One strategic trick, learned in advanced gunnery class, is to use a missile to destroy a parachute by careful aiming, thus causing the robot to plummet to Earth, destroying any robots that happen to be beneath it.

This is really the only method of destroying robots that are already on the ground. The enemy has split the attack into levels, and each level is faster and fiercer than the previous one. Duty calls, so plug your joystick into port 1 and prepare to defend your home as the attack commences! Good luck.

Programming information.

Each procedure is commented with a brief description of what it does. Some of the procedures illustrate interesting programming tricks, however, and I want to expand on them.

The first is the procedure Title(). As stated, it prints the title screen. Notice that it checks the location of the vertical scan VCOUNT and puts color information directly into the hardware registers COLPF0-ColPF3. This causes each scan line on the TV screen to be drawn in a different color. Action! is so fast that you can do this without resorting to machine language display list interrupts.

By using the built-in jiffy timer (RTCLOCK), which advances by one each time a new screen is drawn, in the equation to compute what color is actually used, the colors are made to "scroll" up the screen, providing a rather neat effect. The speed of the scroll is determined by the RSH portion of the color term.

RSH essentially does a divide, so the more times you RSH the RTCLOCK, the slower the scroll will be.

The other interesting procedure is MoveTroopers(), which moves the robots down the screen. As you can see by looking at the program listing, Sneak Attack is written in graphics 0, with a redefined character set. Yet the robots scroll smoothly down.

The way it works is this: each robot is two characters high (chute and robot) and is initially put on the screen by simply printing three characters one above the other—the two characters which make up the shape and a third character which is initially blank. These three characters appear one after the other in the character set.

To move the robot in what looks like a smooth scroll down the screen, the 16 bytes which make up the shape (two characters at 8 bytes per character) are shifted 1 byte further into the 24 bytes of the three characters which were printed on the screen.

This "dynamic character redefinition" goes on until the figure has been shifted 8 bytes down, at which time the top character of the three is blank, and the 16-byte figure resides in the lower two characters. Then, you move the 16 bytes back into the top two characters, and then print the three characters one position lower on the screen.

The shifting of 16 byte blocks is done using MOVEBLOCK. The location of the character set and the location of the 16 bytes which make up the shape are passed to MOVEBLOCK by using the names of the arrays which contain the data. Used in this manner, array names are treated as the memory addresses of the data in the array.

Sneaking around.

I've been programming Atari home computers for four years. The very first video game I ever saw running on a home computer was a little something from Sierra (then Online), called Sabotage. It was only available on the Apple and was never translated.

I've always enjoyed Sabotage and several times tried to program something similar myself. I was never very successful, mostly because BASIC just isn't up to the job. But Action! is, and I think you'll enjoy this version of a venerable game.

One more thing. The end is worth waiting for.

David Plotkin is a Project Engineer for Chevron U.S.A., with a Master's in Chemical Engineering. He bought his Atari in 1980 and is interested in programming and design of games, as well as word processing. His work has been seen in ANALOG Computing, Compute! and other computer magazines.
Listing 1.

; Sneak Attack by David Plotkin

MODULE

BYTE
ChrBase=756, Max, Bkgnd=710,
Fate=$3770, Level=11, CursIn=752,
Stick=362, Ps, Loud=10, Indx=101,
DownL=10, DownR=10, Loud=10,
Snd1=DB200, Snd2=5200, Freq=1191,
Wsync=$D400, Colbk=$D400,
Mnem=$D9F0, Hard=115,
Conso1=53279

CARD
Spin=88, RamSet, HiMem=$2E5,
Score=10, Comp=1000, 5dist=560,
Vdsist=512

CARD ARRAY Linept(24)

BYTE ARRAY
Charset, Chopperstatus(30),
Chopperx(30), Choppery(30),
ExpX(60), ExpY(60), ExpStatus(60),
TrStatus(30), Try(30), Tryst(30),
Missstatus(30), Missx(30), Misy(30),
L1(20), Rr(20), Dist,
ShapeTable(0)=%
1254 16 124 71 127 12 62 0
127 8 62 226 254 24 126 0
96 96 48 40 24 68 231 255
24 24 24 24 24 68 231 255
5 5 12 12 24 68 231 255
128 65 17 66 24 128 91 131
60 126 255 255 192 66 36 24
60 36 24 255 60 24 24 102
0 0 0 0 0 0 0 0
60 36 24 255 60 24 24 102
60 36 24 255 60 24 24 102
60 60 24 255 60 24 24 102
60 60 24 255 60 24 24 195

PROC Update()
; Print score and level
  Position(1,23)
  Print("Score:", Position(8,23)
  PrintC(Score)
  Position(18,23)
  Print("Level: ")
  Position(25,23)
  PrintB(Leve1)
RETURN

PROC Title()
BYTE colpf0=53279, colpf1=53271,
colpf3=53273, colpf2=53271,
rtclock=28, vcount=54283
Graphics(18)
Position(3,4)
Print(6,"SNEAK ATTACK")
Position(6,5)
Print(6,"BY")
Position(3,7)
Print(6,"david plotkin")
Position(3,9)
Print(6,"HEAVY START")
WHILE Consol1<6 DO
  colpf3=Fatc
  Wsync=0
  colpf0=128-vcount+rtclock RSH 2
  colpf1=4count+rtclock RSH 2
  od
RETURN

PROC G8Init()
; Set up the address of each screen
; line and initialize
CARD xx
  Graphics(0)
  Cursin=1
  Print(""
  FOR xx=0 TO 23 DO
    Linept(xxx)=5crn+(40*xx)
  od
  FOR xx=0 TO 29 DO
    Chopperstatus(xx)=0
    Chopperx(xx)=0
    Choppery(xx)=0
    Missx(xx)=0
    Misy(xx)=0
    Missstatus(xx)=0
    Trstatus(xx)=0
  od
  FOR xx=0 TO 59 DO
    Expstatus(xx)=0
  od
  FOR xx=0 TO 19 DO
    L1xx=0
    Rrxx=0
  od
  Bkgnd=0
  Update0
RETURN

PROC Plot8(BYTE x,y, ch)
; Plot a char at location x,y
BYTE ARRAY line
  line=Linept(y)
  line(x)=ch
RETURN

BYTE FUNC Locate8(BYTE x,y)
;Returns the value of the char at x,y
BYTE ARRAY line
  line=LineEmpty()
RETURN(line(x))

PROC Noise()
;the explosion noises
  IF Loud=0 AND Loudi=0
    AND Freq=169 THEN
    RETURN
  FI
  IF Loud THEN
    Loud=-2
    Sound(0,90,8,Loud)
  FI
  IF Loudi THEN
    Loudi=-2
    Sound(1,150,8,Loudi)
  FI
  IF Freq<168 THEN
    Freq+=8
    Sound(2,Freq,10,4)
  ELSE
    Freq=169
    Sound(2,0,0,0)
  FI
RETURN

PROC HitChute BYTE wh
;see which chute was hit by Missile wh
BYTE Ip
FOR Ip=0 TO 29 DO
  IF Misx(wh)=Trx(Ip) AND
     (Misy(wh)=Try(Ip)+1 OR
      Misy(wh)=Try(Ip)+2) THEN
    TrStatus(Ip)=3
    Plot(Trx(Ip),Try(Ip)+1,6)
    Plot(Trx(Ip),Try(Ip)+2,0)
    EXIT
  FI
OD
IF Try(Ip) LSH 3 (< Freq THEN
  Freq=Try(Ip) LSH 3
FI
RETURN

PROC HitMan BYTE wh
;see which man was hit by Missile wh
BYTE Ip
FOR Ip=0 TO 29 DO
  IF Misx(wh)=Trx(Ip) AND
     (Misy(wh)=Try(Ip)+1 OR
      Misy(wh)=Try(Ip)+2) THEN
    TrStatus(Ip)=1
    Plot(Trx(Ip),Try(Ip)+1,0)
    Plot(Trx(Ip),Try(Ip)+2,0)
  FI
OD
IF Loudi=12
RETURN

PROC ExplodeChopper BYTE Ip
;explosions in place of Chopper Ip
BYTE Ip
FOR Ip=0 TO 59 STEP 2 DO ;find empty
  IF ExpStatus(Ip)=0 THEN
    ExpStatus(Ip)=1
    ExpStatus(Ip+1)=1
    Exp(Ip)=Choppery(Ip)
    Exp(Ip+1)=Choppery(Ip)
    Exp(Ip)=Choppery(Ip+1)
    Exp(Ip+1)=Choppery(Ip+1)
  FI
OD
RETURN

Chopperstatus(Ip)=0
Plot(Exp(Ip),Exp(Ip)+1,6)
Plot(Exp(Ip+1),Exp(Ip+1)+1,6)
EXIT
OD
RETURN

PROC HitChopper BYTE wh
;which chopper was hit by Missile Wh
BYTE Ip
FOR Ip=0 TO 29 DO
  IF Misy(wh)=Choppery(Ip) AND
     (Misx(wh)=Chopperx(Ip) OR
      Misx(wh)=Chopperx(Ip)+1) THEN
    ExplodeChopper(Ip)
    EXIT
  FI
OD
RETURN

PROC MissileHit BYTE wh
;see if Missile wh hit anything
BYTE dum
  dum=Locate0(Misx(wh),Misx(wh))
  IF dum=0 THEN
    Plot(Misx(wh),Misy(wh),84)
  RETURN
  FI
  MissStatus(wh)=0
  IF dum=1 OR dum=2 THEN
    HitChopper(wh)
    Score=-1
  ELSEIF (dum=7 AND Indx<6 OR
          dum=8 AND Indx=3) THEN
    HitChute(wh)
    Score=-2
  ELSEIF (dum=8 AND Indx<4 OR
          dum=9 AND Indx=1) THEN
    HitMan(wh)
    Score=-1
  FI
RETURN

PROC Modify
;Modify the RAM character set
CARD xx
  FOR xx=0 TO 127 DO
    Charset(xx+8)=ShapeTable(xx)
  OD
RETURN

PROC LaunchTrooper BYTE wh
;drop a paratrooper from chopper wh
BYTE Ip
  IF Fate<240-(Level LSH 1) THEN
    FOR Ip=0 TO 29 DO ;find MT trooper
      IF TrStatus(Ip)=0 THEN ;got one
        TrStatus(Ip)=1
        Trx(Ip)=Choppery(wh)
      IF Trx(Ip)=0 THEN
        Trx(Ip)=1
      FI
      Try(Ip)=Choppery(Ip+1)
      Plot(Trx(Ip),Try(Ip),7)
      Plot(Trx(Ip),Try(Ip)+1,8)
      Plot(Trx(Ip),Try(Ip)+2,9)
    EXIT
  FI
OD
RETURN

PROC EraseChopper BYTE wh
Erase Chopper Number Wh
Plot0(Chopperx(Wh),Choppery(Wh),0)
Plot0(Chopperx(Wh)+1,Choppery(Wh),0)
Chopperstatus(Wh)=0
Chopperx(Wh)=0
Choppery(Wh)=0
RETURN

PROC DrawChopper(BYTE wh)
;draw chopper number Wh
Plot0(Chopperx(Wh),Choppery(Wh),1)
Plot0(Chopperx(Wh)+1,Choppery(Wh),2)
RETURN

PROC ClearScreen()
;clear the screen
BYTE ip
FOR ip=0 TO 29 DO
  IF Chopperstatus(ip)=1 THEN
    EraseChopper(1p)
  ELSE
    IF Trstatus(ip)=1 THEN
      Trstatus(1p)=0
      Plot0(Trx(ip),Try(ip),0)
      Plot0(Trx(ip),Try(ip)+1,0)
      Plot0(Trx(ip),Try(ip)*2,0)
    FI
    IF Misstatus(ip)=1 THEN
      Misstatus(ip)=0
      Plot0(Misx(ip),Misy(ip),0)
    FI
  OD
FOR ip=0 TO 59 STEP 2 DO
  IF Expstatus(ip)=1 THEN
    Expstatus(ip)=0
    Plot0(Expx(ip),Expy(ip),0)
    Plot0(Expx(ip)+1,Expy(ip),0)
  FI
OD
RETURN

PROC MoveChopper()
;move the choppers
BYTE ip,ps=0
FOR ip=0 TO 29 DO
  IF Chopperstatus(ip)=1 THEN
    Chopperx(ip)=38
    EraseChopper(1p)
  ELSE
    Plot0(Chopperx(ip),Choppery(ip),0)
    Chopperx(ip)=ip+1
    DrawChopper(1p)
    LaunchTrooper(1p)
  FI
  IF Chopperstatus(ip)=2 THEN
    Chopperx(ip)=0
    EraseChopper(1p)
  ELSE
    Plot0(Chopperx(ip)+1,Choppery(ip),0)
    Chopperx(ip)=ip-1
    DrawChopper(1p)
    LaunchTrooper(1p)
  FI
OD
RETURN

PROC Shoot()
;send off a bullet
BYTE Trig=644,lp,fig=0
IF Trig=1 OR fig=0 THEN
  RETURN
  IF Chopperstatus(lp)=0 THEN
    Misstatus(lp)=1
    Misx(lp)=20
    ELSE IF Ps=5 THEN
    Misx(lp)=21
    ELSE
    Misx(lp)=20
    FI
    MissileHit(lp)
    EXIT
    FI
    OD
    fig=0
    RETURN

PROC LaunchChopper()
;decide whether to send off a new
;chopper, which side, how high up
BYTE lp
IF Fate=2300-LEVEL LSH 1 THEN
  FOR lp=0 TO 29 DO
    IF Chopperstatus(lp)=0 THEN
      Chopperx(lp)=Rand(Hard)
      IF Fate=120 THEN
        Chopperx(lp)=38
        Chopperstatus(lp)=1
      ELSE
        Chopperx(lp)=0
        Chopperstatus(lp)=1
      FI
      DrawChopper(lp)
      EXIT
    FI
  OD
RETURN

PROC DrawBase()
;draw the base
BYTE lp
FOR lp=19 TO 21 DO
  Plot0(lp,22,128)
OD
Plot0(20,21,4)
RETURN

PROC AimGun()
;read the joystick and move the base
IF Stick=11 THEN
  Ps=3
ELSEIF Stick=7 THEN
  Ps=5
ELSE
  Ps=4
FI
Plot0(20,21,Ps)
RETURN

PROC Shoot()
;send off a bullet
BYTE Trig=644,lp,fig=0
IF Trig=1 OR fig=0 THEN
  RETURN
  IF Chopperstatus(lp)=0 THEN
    Misstatus(lp)=1
    Misx(lp)=20
    ELSE IF Ps=5 THEN
    Misx(lp)=21
    ELSE
    Misx(lp)=20
    FI
    MissileHit(lp)
    EXIT
    FI
    OD
    fig=0
    RETURN

(Listing continued
on page 60)
PROC MoveShots()
; move the fired bullets
BYTE lp
FOR ip=0 TO 29 DO ; for each shot
IF MissStatus(lp)=1 THEN
Plot0(Missx(lp),Missy(lp),0)
ELSEIF Stick=7 THEN
Missx(lp)=+1
ELSEIF Stick=-1 THEN
Missx(lp)=-1
FI
IF (Missy(lp)<39 AND Missy(lp)>255 AND MissStatus(lp)=0) THEN
MissileHit(lp)
ELSE
MissStatus(lp)=0
FI
OD
RETURN

PROC MoveExplosions()
; move the explosions
BYTE lp
FOR ip=0 TO 59 STEP 2 DO
IF ExpStatus(lp)=1 THEN
Plot0(Expx(lp),Expy(lp),0)
Plot0(Expx(lp)+1,Expy(lp)+1,0)
Expx(lp)=+1
Expy(lp)=-1
Expx(lp)=+1
Expy(lp)=-1
IF Expx(lp)<22 AND Expx(lp)<0 AND Expy(lp)<39 THEN
Plot0(Expx(lp),Expy(lp)+1,0)
Plot0(Expx(lp)+1,Expy(lp)+1,0)
ELSE
ExpStatus(lp)=0
ExpStatus(lp+1)=0
FI
FI
OD
RETURN

PROC BaseExplode()
; explode the base
BYTE ARRAY endx(0)=16 24 17 23 20),
endy(0)=22 22 19 19 17)
BYTE lp, time=20
COLOR=38
FOR ip=0 TO 4 DO
Plot0(20,22)
DrawTo(endx(ip),endy(ip))
OD
FOR ip=0 TO 16 DO
Sound(0,Fate,8,16-lp)
Sound(1,Fate,8,16-lp)
time=0
DO
UNTIL time=15
OD
OD
ShndRst()
COLOR=32
FOR ip=0 TO 4 DO
Plot0(20,22)
DrawTo(endx(ip),endy(ip))
OD
RETURN

PROC EndLeft()
; move the troops from the left to the base
BYTE lp, lc, nn, time=20
FOR ip=0 TO 19 DO
lp=19-lp
IF L1(lq)=1 THEN
FOR lc=1q TO 19 DO
IF nn=12 THEN
ELSE
nn=12
FI
Plot0(lc,22,nn)
time=0
DO
UNTIL time=10
OD
Plot0(lc,22,0)
OD
Plot0(19,22,11)
FI
OD
FOR lp=0 TO 3 DO
Plot0(21,221p,11)
time=0
DO
UNTIL time=10
OD
OD
BaseExplode()
RETURN

PROC EndRight()
; move the troops from the right
Print("Press FIRE to play again")
DO
  UNTIL trig=0
  DOWNL=0
  DOWNR=0
  Put(125)
  FOR Ip=0 TO 19
    Li(Ip)=0
    Rr(Ip)=0
  OD
  Score=0
  Level=1
  DrawBase()
  Update()
  Hard=15
RETURN

PROC TrooperFall1()
; make trooper fall when chute hit
BYTE Ip, qq, cc
FOR Ip=0 TO 29
  IF TrStatus(Ip)=2 THEN
    Plot0(Ip, Trx(Ip), Try(Ip)+1, 10)
    Try(Ip)=4
    IF Try(Ip)=21 THEN
      cc=Trx(Ip)
      IF Trx(Ip)<20 AND Li(cc)=1 THEN
        DownL=-1
        Li(cc)=0
      ELSEIF Trx(Ip)>20 AND Rr(cc-21)=1 THEN
        DownR=-1
      FI
    FI
    IF (Try(Ip)<22 AND Trx(Ip)>20) OR (Try(Ip)>20 AND Trx(Ip)<20) THEN
      Plot0(Ip, Try(Ip)+1, 10)
    ELSE
      TrStatus(Ip)=0
    FI
  OD
RETURN

PROC GameOverTwo()
; game over when four troopers down
BYTE Ip
  SndRst()
  ClearScreen()
  Loud=0
  Loudi=0
  Freq=169
FOR Ip=0 TO 19
  IF Li(Ip)=1 THEN
    Plot0(Ip, 22, 11)
  FI
  IF Rr(Ip)=1 THEN
    Plot0(Ip+21, 22, 11)
  FI
  IF DownL=4 THEN
    EndLeft()
  ELSE
    EndRight()
  FI
  EndPrint()
RETURN

PROC GameOverOne()
; game over when trooper lands on base
BYTE Ip
  SndRst()
  ClearScreen()
  Loud=0
  Loudi=0
  Freq=169
FOR Ip=0 TO 19
  IF Li(Ip)=1 THEN
    Plot0(Ip, 22, 11)
  FI
  IF Rr(Ip)=1 THEN
    Plot0(Ip+21, 22, 11)
  FI
  IF BaseExplode() THEN
    Exit()
  FI
END
RETURN

PROC TrooperDown(BYTE wh)
; redraw trooper wh at bottom of screen
BYTE cc
  TrStatus(wh)=0
  cc=Trx(wh)
  Plot0(Trx(wh), Try(wh), 0) ; erase chute
  Plot0(Trx(wh), Try(wh)+1, 11) ; place
  IF Trx(wh)<20 AND Li(cc)=0 THEN
    Li(cc)=1
    DownL=-1
  ELSEIF Trx(wh)>=20 AND Rr(cc-21)=0 THEN
    DOWNR=-1
  FI
RETURN

PROC MoveTroopers()
; move paratroopers down screen
BYTE Ip, qq
BYTE ARRAY Trooper(0)=
  [60 126 255 195 66 36 24 60 36 24 255 60 24 16 102 0 0 0 0 0 0 0]
FOR Ip=0 TO Indx DO
  Charset56+Ip=0
  MoveBlock(Charset+56+Ip+1, Trooper, 16)
  Indx+=1
  IF Indx<0 THEN
    RETURN
  FI
END
RETURN

PROC MoveBlock(Charset+56, Trooper, 24)
FOR Ip=0 TO 29
  IF TrStatus(Ip)=1 THEN
    Plot0(Ip, Trx(Ip), Try(Ip)+1, 10)
    Try(Ip)=4
    IF Try(Ip)=21 THEN
      TrStatus(Ip)=0
    FI
  FI
  IF TrStatus(Ip)=3 THEN
    Plot0(Ip, Trx(Ip), Try(Ip)+1, 10)
  FI
END
RETURN

ANALOG COMPUTING

NOVEMBER 1985 / PAGE 61
**PROC NewLevel()**

```assembly
go to higher level
BYTE lp, time=20
Level+=1
IF Level>100 THEN
    Level=100
FI
FI
5ndRst()
Loud=0
Loud=0
Freq=169
Comp=+380
FOR ip=10 TO 150 STEP 10 DO
    Sound(#ip, #ip, 10, 4)
    Sound(#ip, #ip, 10, 4)
    time=0
    DO
    UNTIL time=2
    OD
    Position(25, 23)
    PrintB(Level)
    IF Level<8 THEN
        Hard=19
    FI
    5ndRst()
    RETURN
```

---

**PARTS / SERVICE FOR ATARI COMPUTERS**

Flat Service Rates Below Include Parts & Labor, 60-Day Warranty

<table>
<thead>
<tr>
<th>Service</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>$150.00</td>
</tr>
<tr>
<td>CPU</td>
<td>$100.00</td>
</tr>
<tr>
<td>Video</td>
<td>$75.00</td>
</tr>
<tr>
<td>Power Supply</td>
<td>$50.00</td>
</tr>
</tbody>
</table>

**BARE BOARDS**

- With parts data: $150.00
- Build your own interface: $150.00
- Basic Kit: $150.00
- Advanced Kit: $200.00
- Power Supply: $50.00
- SAMS Service Manual: $50.00
- De Ha Kit: $100.00
- Inside Basic Kit: $60.00

**SOFTWARE**

- Basic Carnegie: $15.00
- Editor Carnegie: $15.00
- O'Bern Carnegie: $15.00
- PopEye Carnegie: $15.00
- Kindergarten Cart: $10.00
- Buck Rogers Cart: $7.50
- Jumbo Jet Pilot: $10.00
- Crossfire Cart: $5.00
- Chicken Cartridge: $5.00

**AMERICAN TV**  
**415-352-3787**

Mail Order and Repair: 15238 Inverness St., San Leandro, CA 94579
Retail Store: 7831 Washington Avenue, San Leandro, CA 94577
Terms: We accept Money orders, personal checks or C.O.D.'s - VISA, MasterCard okay on orders over $50.00. No personal checks on C.O.D.
Shipping: $4.00 shipping and handling under $100.00. Add $2.00 for C.O.D. orders. California residents include 6% sales tax. Overseas shipping extra.
Prices subject to change without notice. We reserve the right to limit quantities. Sales limited to stock on hand. Foreign shipping extra.

---

**SNEAK ATTACK**

**PASCAL**

*Pascal for the Atari*

kyan pascal is the most complete package available for learning and using the Pascal language. And now, it’s available for the Atari family of computers.

kyan pascal is easy to use, lightning fast, and loaded with features. It’s a full Pascal implementation and perfect for both beginners and advanced programmers.

**kyan pascal features:**
- Optimized Machine Code Compiler
- Built-In Full Screen Text Editor
- String-Handling, Chaining, Include File, and Other Powerful Extensions
- 13 Digit Arithmetic Precision and Full Graphics Support
- Built-In Assembler (Allows Full Hardware Control)
- Tutorial Manual with Sample Programs

**kyan pascal for the Atari**

Add a $10.00 copy of handling. $10.00 outside North America. California.

Send Check/Money Order to: kyan software, Dept M
Call: (415) 775-2923
1850 Union St., Ste 183
MC/Visa Accepted
San Francisco, CA 94123