

General Information

Author: David Plotkin

Language: ACTION!

Compiler/Interpreter: ACTION!

Published: Analog #36 (11/ 85)

Sneak Attack#

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.*

```
; Sneak Attack by David Plotkin
```

```
MODULE
```

```
BYTE
```

```
ChrBase=756,Max,Bkgrnd=710,  
Fate=53770,Level=[1],CursIn=752,  
Stick=632,Ps,Loud=[0],Indx=[0],  
DownL=[0],DownR=[0],Loud1=[0],  
Snd1=$D208,Snd2=$D20F,Freq=[169],  
Wsync=$D40A,Colbk=$D018,  
Nmien=$D40E,Hard=[15],  
Consol=53279
```

CARD

```
Scrn=88,RamSet,HiMem=$2E5,  
Score=[0],Comp=[300],Sdlst=560,  
Vdslst=512
```

CARD ARRAY Linept(24)

BYTE ARRAY

```
Charset,Chopperstatus(30),  
Chopperx(30),Choppery(30),  
Expx(60),Expy(60),ExpStatus(60),  
TrStatus(30),Trx(30),Try(30),  
MisStatus(30),Misx(30),Misy(30),  
Ll(20),Rr(20),Dlist,  
ShapeTable(0)=  
[254 16 124 71 127 12 62 0  
127 8 62 226 254 24 126 0  
96 96 48 48 24 60 231 255  
24 24 24 24 24 60 231 255  
6 6 12 12 24 60 231 255  
128 85 17 66 24 170 91 131  
60 126 255 255 195 66 36 24  
60 36 24 255 60 24 36 102  
0 0 0 0 0 0 0 0  
60 36 24 255 60 24 36 102  
60 36 219 255 60 24 36 102  
60 60 24 60 60 24 24 28  
60 60 24 60 60 60 102 195]
```

PROC Download()

```
;Step back HiMem and move the  
;character set into RAM
```

CARD Index

BYTE Val

```
RamSet=(HiMem-$400)&$FC00  
ChrBase=RamSet RSH 8  
HiMem=RamSet  
FOR Index=0 TO 1023 DO  
Val=Peek(57344+Index)  
Poke(RamSet+Index,Val)  
OD  
Charset=RamSet
```

RETURN

PROC Dlint()

```
;the display list interrupt routine  
[$48 $8A $48 $98 $48]  
Wsync=1  
Colbk=50  
[$68 $A8 $68 $AA $68 $40]
```

PROC ScoreLine()

```
;set up the dli  
Dlist=Sdlst  
Vdslst=Dlint
```

```
Dlist(27)=130
Nmien=$C0
RETURN
```

```
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(Level)
RETURN
```

```
PROC Title()
BYTE colpf0=53270,colpf1=53271,
     colpf2=53273,colpf3=53273,
     rtclock=20,vcount=54283
Graphics(18)
Position(3,4)
PrintD(6,"SNEAK ATTACK")
Position(8,5)
PrintD(6,"BY")
Position(3,7)
PrintD(6,"david plotkin")
Position(3,9)
PrintD(6,"PRESS start")
WHILE Consol<>6 DO
  colpf3=Fate
  Wsync=0
  colpf0=128-vcount+rtclock RSH 2
  colpf1=vcount+rtclock RSH 2
OD
RETURN
```

```
PROC Gr0Init()
;Set up the address of each screen
;line and initialize
CARD xx
Graphics(0)
CursIn=1
Print(" ")
FOR xx=0 TO 23 DO
  Linept(xx)=Scrn+(40*xx)
OD
FOR xx=0 TO 29 DO
  Chopperstatus(xx)=0
  Chopperx(xx)=0
  Choppery(xx)=0
  Misx(xx)=0
  Misy(xx)=0
  MisStatus(xx)=0
  TrStatus(xx)=0
OD
```

```

FOR xx=0 TO 59 DO
  ExpStatus(xx)=0
OD
FOR xx=0 TO 19 DO
  Ll(xx)=0
  Rr(xx)=0
OD
Bkgrnd=0
Update()
RETURN

```

```

PROC Plot0(BYTE x,y,ch)
;Plot a char at location x,y
BYTE ARRAY line
  line=Linept(y)
  line(x)=ch
RETURN

```

```

BYTE FUNC Locate0(BYTE x,y)
;Returns the value of the char at x,y
BYTE ARRAY line
  line=Linept(y)
RETURN(line(x))

```

```

PROC Noise()
;the explosion noises
  IF Loud=0 AND Loud1=0
    AND Freq=169 THEN
    RETURN
  FI
  IF Loud THEN
    Loud== -2
    Sound(0,90,8,Loud)
  FI
  IF Loud1 THEN
    Loud1== -2
    Sound(1,150,8,Loud1)
  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 lp
  FOR lp=0 TO 29 DO
    IF Misx(wh)=Trx(lp) AND
      (Misx(wh)=Try(lp) OR
        Misx(wh)=Try(lp)+1) THEN
      TrStatus(lp)=2
    ENDIF
  ENDFOR

```

```

        Plot0(Trx(lp),Try(lp),0)
        Plot0(Trx(lp),Try(lp)+1,10)
        Plot0(Trx(lp),Try(lp)+2,0)
    EXIT
FI
OD
IF Try(lp) LSH 3 < Freq THEN
    Freq=Try(lp) LSH 3
FI
RETURN

```

```

PROC HitMan(BYTE wh)
;see which man was hit by missile wh
BYTE lp
    FOR lp=0 TO 29 DO
        IF Misx(wh)=Trx(lp) AND
            (Misy(wh)=Try(lp)+1 OR
            Misy(wh)=Try(lp)+2) THEN
            TrStatus(lp)=3
            Plot0(Trx(lp),Try(lp)+1,6)
            Plot0(Trx(lp),Try(lp),0)
            Plot0(Trx(lp),Try(lp)+2,0)
        FI
    OD
    Loud1=12
RETURN

```

```

PROC ExplodeChopper(BYTE lp)
;explosions in place of Chopper lp
BYTE lq
    FOR lq=0 TO 59 STEP 2 DO ;find empty
        IF ExpStatus(lq)=0 THEN
            ExpStatus(lq)=1
            ExpStatus(lq+1)=1
            ExpX(lq)=ChopperX(lp)
            ExpX(lq+1)=ChopperX(lp)+1
            ExpY(lq)=ChopperY(lp)
            ExpY(lq+1)=ChopperY(lp)
            ChopperStatus(lp)=0
            Plot0(ExpX(lq),ExpY(lq),6)
            Plot0(ExpX(lq+1),ExpY(lq+1),6)
        EXIT
    FI
OD
RETURN

```

```

PROC HitChopper(BYTE wh)
;which chopper was hit by missile wh
BYTE lp
    FOR lp=0 TO 29 DO
        IF Misy(wh)=ChopperY(lp) AND
            (Misx(wh)=ChopperX(lp) OR
            Misx(wh)=ChopperX(lp)+1) THEN
            ExplodeChopper(lp)
        EXIT
    FI
OD

```

Loud=12

RETURN

PROC MissileHit(BYTE wh)

;see if missile wh hit anything

BYTE dum

dum=Locate0(Misx(wh),MisY(wh))

IF dum=0 THEN

Plot0(Misx(wh),MisY(wh),84)

RETURN

FI

MisStatus(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 103 DO

Charset(xx+8)=ShapeTable(xx)

OD

RETURN

PROC LaunchTrooper(BYTE wh)

;drop a paratrooper from chopper wh

BYTE lp

IF Fate>240-(Level LSH 1) THEN

FOR lp=0 TO 29 DO ;find MT trooper

IF TrStatus(lp)=0 THEN ;got one

TrStatus(lp)=1

Trx(lp)=Chopperx(wh)

IF Trx(lp)=0 THEN

Trx(lp)=1

FI

Try(lp)=Choppery(wh)+1

Plot0(Trx(lp),Try(lp),7)

Plot0(Trx(lp),Try(lp)+1,8)

Plot0(Trx(lp),Try(lp)+2,9)

EXIT

FI

OD

FI

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 lp
  FOR lp=0 TO 29
  DO
    IF Chopperstatus(lp) THEN
      EraseChopper(lp)
    FI
    IF TrStatus(lp) THEN
      TrStatus(lp)=0
      Plot0(Trx(lp),Try(lp),0)
      Plot0(Trx(lp),Try(lp)+1,0)
      Plot0(Trx(lp),Try(lp)+2,0)
    FI
    IF MisStatus(lp)=1 THEN
      MisStatus(lp)=0
      Plot0(Misx(lp),Misy(lp),0)
    FI
  OD
  FOR lp=0 TO 59 STEP 2 DO
    IF ExpStatus(lp)=1 THEN
      ExpStatus(lp)=0
      ExpStatus(lp+1)=0
      Plot0(ExpX(lp),ExpY(lp),0)
      Plot0(ExpX(lp+1),ExpY(lp+1),0)
    FI
  OD
RETURN

```

```

PROC MoveChopper()
;move the choppers
BYTE lp,ps=[0]
  FOR lp=0 TO 29 DO
    IF Chopperstatus(lp)=1 THEN ;right
      IF Chopperx(lp)=38 THEN
        EraseChopper(lp)
      ELSE
        Plot0(Chopperx(lp),
              Choppery(lp),0)
        Chopperx(lp)==+1
        DrawChopper(lp)
        LaunchTrooper(lp)
      FI
    FI
  OD

```

```

    FI
  FI
  IF Chopperstatus(lp)=2 THEN ;left
    IF Chopperx(lp)=0 THEN
      EraseChopper(lp)
    ELSE
      Plot0(Chopperx(lp)+1,
            Chopperry(lp),0)
      Chopperx(lp)=-1
      DrawChopper(lp)
      LaunchTrooper(lp)
    FI
  FI
  FI
  OD
  IF ps=0 THEN
    Charset(8)=56
    Charset(16)=28
    ps=1
  ELSE
    ps=0
    Charset(8)=254
    Charset(16)=127
  FI
  RETURN

```

```

PROC LaunchChopper()
;Decide whether to send off a new
;chopper, which side, how high up
BYTE lp
  IF Fate>230-(Level LSH 1) THEN
    FOR lp=0 TO 29 DO ;find MT chopper
      IF Chopperstatus(lp)=0 THEN
        Chopperry(lp)=Rand(Hard)
        IF Fate>128 THEN
          Chopperx(lp)=38 ;right side
          Chopperstatus(lp)=2
        ELSE
          Chopperx(lp)=0 ;left side
          Chopperstatus(lp)=1
        FI
        DrawChopper(lp)
        EXIT
      FI
    OD
  FI
  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,flg=[0]
  IF trig=1 OR flg=0 THEN
    flg=1
    RETURN
  FI
  FOR lp=0 TO 29 DO ;find empty shot
    IF MisStatus(lp)=0 THEN ;got one
      MisStatus(lp)=1
      Misy(lp)=20
      IF Ps=3 THEN
        Misx(lp)=19
      ELSEIF Ps=5 THEN
        Misx(lp)=21
      ELSE
        Misx(lp)=20
      FI
      MissileHit(lp)
      EXIT
    FI
  OD
  flg=0
RETURN

```

```

PROC MoveShots()
;move the fired bullets
BYTE lp
  FOR lp=0 TO 29 DO ;for each shot
    IF MisStatus(lp)=1 THEN
      Plot0(Misx(lp),Misy(lp),0)
      IF Stick=11 THEN
        Misx(lp)==-1
      ELSEIF Stick=7 THEN
        Misx(lp)==+1
      ELSE
        Misy(lp)==-1
      FI
      IF (Misx(lp)<>39 AND
        Misy(lp)<>255 AND
        Misx(lp)<>0) THEN
        MissileHit(lp)
      ELSE
        MisStatus(lp)=0
      FI
    FI
  OD

```

```
    FI
  OD
RETURN
```

```
PROC MoveExplosions()
;move the explosions
BYTE lp
  FOR lp=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)
      Expy(lp)==+1
      Expy(lp+1)==+1
      ExpX(lp)==-1
      ExpX(lp+1)==+1
      IF Expy(lp)<>22 AND ExpX(lp)<>0
        AND ExpX(lp+1)<>39 THEN
        Plot0(ExpX(lp),ExpY(lp),6)
        Plot0(ExpX(lp+1),ExpY(lp+1),6)
      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 lp=0 TO 4 DO
    Plot(20,22)
    DrawTo(endx(lp),endy(lp))
  OD
  FOR lp=0 TO 16 DO
    Sound(0,Fate,8,16-lp)
    Sound(1,Fate,8,16-lp)
    time=0
    DO
      UNTIL time=15
    OD
  OD
  SndRst()
  color=32
  FOR lp=0 TO 4 DO
    Plot(20,22)
    DrawTo(endx(lp),endy(lp))
  OD
RETURN
```

```
PROC EndRight()
;move the troopers from the right
;to the base
```

```

BYTE lp,lq,nn,time=20
FOR lp=0 TO 19 DO
  IF Rr(lp)=1 THEN
    lq=21+lp
    WHILE lq>20 DO
      IF nn=12 THEN
        nn=13
      ELSE
        nn=12
      FI
      Plot0(lq,22,nn)
      time=0
      DO
        UNTIL time=10
      OD
      Plot0(lq,22,0)
      lq=-1
    OD
    Plot0(21,22,11)
  FI
OD
FOR lp=0 TO 3 DO
  Plot0(21,22-lp,11)
  time=0
  DO
    UNTIL time=10
  OD
OD
BaseExplode()
RETURN

```

```

PROC EndLeft()
;Move the troopers from the left to
;the base
BYTE lp,lq,lc,nn,time=20
FOR lp=0 TO 19 DO
  lq=19-lp
  IF Ll(lq)=1 THEN
    FOR lc=lq TO 19 DO
      IF nn=12 THEN
        nn=13
      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(19,22-lp,11)
  time=0
  DO

```

```
        UNTIL time=10
    OD
    OD
    BaseExplode()
RETURN
```

```
PROC EndPrint()
;print the end of game message and
;test for new game
BYTE trig=644,lp
    Position(10,7)
    Print("Game Over...Final Score:")
    Position(15,8)
    PrintC(Score)
    Position(15,9)
    Print("FINAL LEVEL :")
    PrintB(Level)
    Position(10,20)
    Print("Press FIRE to play again")
    DO
        UNTIL trig=0
    OD
    DownL=0
    DownR=0
    Put(125)
    FOR lp=0 TO 19 DO
        Ll(lp)=0
        Rr(lp)=0
    OD
    Score=0
    Level=1
    DrawBase()
    Update()
    Hard=15
RETURN
```

```
PROC GameOverTwo()
;game over when four troopers down
BYTE lp
    SndRst()
    ClearScreen()
    Loud=0
    Loud1=0
    Freq=169
    FOR lp=0 TO 19 DO
        IF Ll(lp)=1 THEN
            Plot0(lp,22,11)
        FI
        IF Rr(lp)=1 THEN
            Plot0(lp+21,22,11)
        FI
    OD
    IF DownL=4 THEN
        EndLeft()
    ELSE
        EndRight()
    FI
```

```
EndPrint()  
RETURN
```

```
PROC GameOverOne()  
;game over when trooper lands on base  
BYTE lp  
  SndRst()  
  ClearScreen()  
  Loud=0  
  Loud1=0  
  Freq=169  
  FOR lp=0 TO 19 DO  
    IF Ll(lp)=1 THEN  
      Plot0(lp,22,11)  
    FI  
    IF Rr(lp)=1 THEN  
      Plot0(lp+21,22,11)  
    FI  
  OD  
  BaseExplode()  
  EndPrint()  
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) ;replace  
  IF Trx(wh)<20 AND Ll(cc)=0 THEN  
    Ll(cc)=1  
    DownL==+1  
  ELSEIF Trx(wh)>20 AND  
    Rr(cc-21)=0 THEN  
    Rr(cc-21)=1  
    DownR==+1  
  ELSEIF Trx(wh)=20 THEN  
    GameOverOne()  
  FI  
  IF DownL=4 OR DownR=4 THEN  
    GameOverTwo()  
  FI  
RETURN
```

```
PROC TrooperFall()  
;make trooper fall when chute hit  
BYTE lp,qq,cc  
  FOR lp=0 TO 29 DO  
    IF TrStatus(lp)=2 THEN  
      Plot0(Trx(lp),Try(lp)+1,0)  
      Try(lp)==+1  
      IF Try(lp)=21 THEN  
        cc=Trx(lp)  
        IF Trx(lp)<20 AND Ll(cc)=1 THEN  
          DownL==-1  
        FI  
      FI  
    FI  
  OD
```

```

        Ll(cc)=0
    ELSEIF Trx(lp)>20 AND
        Rr(cc-21)=1 THEN
        Rr(cc-21)=0
        DownR== -1
    FI
FI
IF (Try(lp)<22 AND Trx(lp)<>20)
    OR (Try(lp)<20 AND
        Trx(lp)=20) THEN
    Plot0(Trx(lp),Try(lp)+1,10)
ELSE
    TrStatus(lp)=0
FI
FI
OD
RETURN

```

```

PROC MoveTroopers()
;move paratroopers down screen
BYTE lp,qq
BYTE ARRAY Trooper(0)=
    [60 126 255 255 195 66 36 24
     60 36 24 255 60 24 36 102
     0 0 0 0 0 0 0 0]
FOR lp=0 TO Indx DO
    Charset(56+lp)=0
OD
MoveBlock(Charset+56+Indx+1,
    Trooper,16)
Indx==+1
IF Indx<8 THEN
    RETURN
FI
Indx=0
FOR lp=0 TO 29 DO
    IF TrStatus(lp)=1 THEN
        Plot0(Trx(lp),Try(lp),0)
        Try(lp)==+1
        IF Try(lp)=21 THEN
            TrooperDown(lp)
        FI
    FI
    IF TrStatus(lp)=3 THEN
        TrStatus(lp)=0
        Plot0(Trx(lp),Try(lp)+1,0)
    FI
OD
MoveBlock(Charset+56,Trooper,24)
FOR lp=0 TO 29 DO
    IF TrStatus(lp)=1 THEN
        Plot0(Trx(lp),Try(lp),7)
        Plot0(Trx(lp),Try(lp)+1,8)
        Plot0(Trx(lp),Try(lp)+2,9)
    FI
OD
RETURN

```

```

PROC NewLevel()
;go to higher level
BYTE lp,time=20
  Level==+1
  IF Level>100 THEN
    Level=100
  FI
  SndRst()
  Loud=0
  Loud1=0
  Freq=169
  Comp==+300
  FOR lp=10 TO 150 STEP 10 DO
    Sound(0,lp,10,4)
    Sound(1,lp+10,10,4)
    time=0
    DO
      UNTIL time=2
    OD
  OD
  Position(25,23)
  PrintB(Level)
  IF Level>8 THEN
    Hard=19
  FI
  SndRst()
RETURN

```

```

PROC Main()
BYTE time=20,lp,ch=764
  Title()
  Gr0Init()
  Snd1=0
  Snd2=3
  Download()
  Modify()
  DrawBase()
  ScoreLine()
  DO
    LaunchChopper()
    MoveChopper()
    MoveExplosions()
    Noise()
    TrooperFall()
    MoveTroopers()
    Position(8,23)
    PrintC(Score)
    IF Score>Comp THEN
      NewLevel()
    FI
    time=0
    FOR lp=2 TO 6 STEP 2 DO
      AimGun()
      Shoot()
      MoveShots()
    DO
      UNTIL time=lp

```

OD

OD

OD

RETURN

PDF: [sneak_attack.pdf](#)