Aunt's in Space

As Autumn approaches and the nights get shorter pistol packin' Aunt John Kennedy shows us how to write that timeless classic – Space Invaders.

After the high score table it seems only fitting that at last we start writing a game. The crusty old classic *Space Invaders* is an excellent first game to write. It has many moving missiles, bombs and bases about which to worry, not to mention the aliens wandering round in formation. Once you get the simplest version of Space Invaders working you can add features to your heart's content.

The source code is so long – around 14K – that we will have to split it into two parts and list the remaining half next month. When assembled, the code takes less than 6K of memory, including all the data needed for the graphics.

The program makes extensive use of a specially-written machine code subroutine to write graphics to the screen. Standard screen writing routines supplied by the firmware are not fast enough to move all the aliens and things round the screen fast enough. Making use of a short table of addresses, the routine can put multi-coloured, character-sized blobs on to the screen very quickly. You are welcome to use the routine in any of your programs but you will have to wait for next month before it is listed.

The first stage in writing any program is an initial design. Everyone knows how Space Invaders works; a block of aliens moves left and right across the screen, dropping bombs on the user's little base. In this version the aliens are in a block of 28, seven columns of four rows. The status of the aliens, either dead or alive, is held in a 28-byte array of memory. The aliens move left and right until they reach the edge of the screen, when they all move down a row. If they are all shot more will appear lower down the screen and, on later levels, rather faster and nastier.

This month's listing includes the main control part of the program, the sections

```
;The following firmware calls are used:

set_mode equ &bc0e

test_key equ &bb1e

txt_output equ &bb5a

set_cursor equ &bb75

wait_frame equ &bd19


org &4000                                  ;Start of Object code. And why not?


START

        call TITLE_SCREEN

        call SET_MAJOR_VARIABLES

MAINLOOP

        call SET_VARIABLES

        call SET_SCREEN

        call SET_SKILL

        call SET_INVADERS

        call WAIT

loop

        call INVADERS                      ;Print & Move Invaders

        call BASE                          ;Print & Move Base

        ld a,(flag)

        cp 0:jr z,loop

        cp 1:jr z,Quit_game

        cp 2:jr z,Game_over

        cp 3:jr z,Lost_a_life

        cp 4:jr z,New_Sheet


Quit_game

        ld a,2:jp Set_mode


New_sheet

        ld hl,string4:call PRINT_STRING

        ld a,(sheet):inc a:ld (sheet),a

        call wait

        jr mainloop


Game_over

        ld hl,string3:call PRINT_STRING

        call WAIT:call WAIT:call WAIT:jp START

Lost_a_life

        ld ix,space:call PRINT_MISSILE                   ;Erase any missiles.

        ld a,0:ld (bomb),a:ld (missile),a:ld (flag),a    ;Reset variables.

        ld a,(lives):dec a:ld (lives),a                  ;Reduce lives.

        call PRINT_LIVES

        call WAIT

        ld a,(lives)

        cp 0                                             ;If some lives left

        jr nz,loop                                       ;then play on.

        jr Game_over


WAIT    ld b,100                                         ;Produce a short pause

wloop   call Wait_Frame

        djnz wloop
```

which define the variables used and the code needed to move the player's base left and right and control the player's missile. The code will be of only academic use until it is combined with next month's listing. When that happens you will have a complete, fully-playable, 100 percent machine code Space Invaders game – always assuming that you have typed it all into an assembler and made no mistakes.

Let us go through the sections and see what is happening. Any labels in capital letters are main subroutines which can be called from any other routine in the program. Labels in lower-case are generally used only from within the section of code in which they are defined. If you are using an assembler other than the MAXAM 1.5 with which the listings were produced you may find that the label names will have to be altered slightly. Some assemblers will refuse the underscore character. Others may limit the length of the label to eight characters. If that is so, change them carefully, preferably taking a pen to the listing in the magazine and jotting down any changes.

The first section of assembler controls the overall structure of the program. It starts the code running at address &4000, although you can change this if necessary. The first thing to happen is that the title screen is drawn. It consists of only a few lines of text and is an obvious candidate for improvement. What about a picture drawn with an art package? If you have a CPC6128 you could store the picture in the extra bank RAM or copy it to the screen when you need it.

The 'major' variables which are next set up are those such as the score and the number of lives with which the player starts. They are set up outside the main loop because they are set only once at the start of each game. Inside the main loop the play variables are set up. They define where the player's base will first appear, how high up the screen the aliens will start and so on; they must be defined at the start of every wave, not just once. The main screen display – score shields – are drawn next and then the final sets of variables controlling the initial skill levels and aliens are set up. After a short pause to allow the player to catch his breath, the game starts in earnest.

```
        ret


; ------------------- Controlling the Variables -----------------------


SET_VARIABLES

        ld iy,aliengr2          ;Aliengr2 points to some graphics data

        ld a,0:ld (animate),a
        ld a,20

        ld (basepos),a          ;Player's base's starting point.

        ld a,0

        ld (flag),a             ;Main flag

        ld (hits),a             ;How many aliens destroyed so far

        ld hl,0

        ld (bomb),hl            ;Two variables; one to control the Aliens

        ld (missile),hl         ;bombs, one to control the players missile

        ret


SET_MAJOR_VARIABLES

        ld hl,0

        ld (score),hl

        ld a,3

        ld (lives),a

        ld a,1

        ld (sheet),a

        ld a,6

        ld (speed),a            ;How fast the aliens move.

        ret

SET_SKILL

        ld a,(sheet):inc a      ;Set the aliens' starting height

        ld (ypos),a             ;depending on the current sheet.

        cp 11

        ret nz

        ld a,10:ld (ypos),a

        ld a,(speed)            ;Speed the aliens up.

        dec a

        ld (speed),a

        cp 0

        ret z

        ld a,1:ld (speed),a

        ret


SET_INVADERS  6

        ld b,28                 ;Set all elements in array.

        ld hl,invadersdata

lpa     ld (hl),1

        inc hl

        djnz lpa

        ld a,2

        ld (xpos),a             ;How far across they start.

        ld a,1

        ld (dir),a              ;Their direction to the left

        ld a,0

        ld (count),a            ;Used to determine their speed.

        ret
```

```
;-------------------- Writing to the Screen --------------------


TITLE_SCREEN                                            ;The bit with my name

        ld a,1:call Set_mode                            ;on it!

        ld hl,string6:call PRINT_STRING

not_pressed

        ld a,47:call Test_key                           ;Wait until Space-bar

        jr z, not_pressed                               ;is pressed.

        ret
SET_SCREEN                                              ;Set the screen up to

        ld a,1:call set_mode                            ;play a game.
        ld hl,string1:call PRINT_STRING

        ld hl,string2:call PRINT_STRING

        call PRINT_SHEET

        call PRINT_SCORE

        call PRINT_LIVES

        call PRINT_HIGH

        ret


PRINT_SCORE

        ld hl,1793:call set_cursor:ld hl,(score):jp printhl


PRINT_HIGH

        ld hl,8705:call set_cursor:ld hl,(high):jp printhl


PRINT_SHEET

        ld hl,5889:call set_cursor:ld a,(sheet):ld l,a:ld h,0:jp printhl


PRINT_LIVES                             ;Print a marker for every life left.

        ld hl,string5:call PRINT_STRING

        ld a,(lives)

        cp 0:ret z

        ld b,a

llop    ld a,239:call txt_output

        djnz llop

        ret
;------------------------------ THE BASE ----------------------------------

BASE    call PRINT_BASE                                 ;Control the player's

        call MOVE_BASE                                  ;base and the missile.

        call CONTROL_MISSILE

        ret




PRINT_BASE                                              ;Print the players base

        ld a,(basepos)

        ld l,a

        ld h,24

        push hl:ld ix,basegr:call PRINT_CHAR:pop hl

        inc l:inc l
        push hl:ld ix,basegr+16:call PRINT_CHAR:pop hl

        inc l:inc l

        ld ix,basegr+32:call PRINT CHAR
        ret

MOVE_BASE                                           ;Check keyboard and
        ld a,66:call Test_key                       ;move player's base
        call nz,esc                                 ;accordingly.
        ld a,(left):call Test_key
        call nz,baseleft
        ld a,(right):call Test_key
        call nz,baseright
        ld a,(fire):call Test_key
        call nz,basefire
        ret
 esc     ld a,1:ld (flag),a                          ;An option to allow
         ret                                         ;a return to BASIC.
 baseleft
         ld a,(basepos)
         cp 0
         ret z                                       ;The the base to the
         dec a                                       ;left.
         ld (basepos),a
         ret
 baseright
         ld a,(basepos)
         cp 70                                       ;Move it to the right.
         ret z
         inc a
         ld (basepos),a
         ret
 basefire
         ld a,(missile)                              ;If a missile is not
         cp 0                                        ;already moving, fire
         ret nz                                      ;one by setting its
         ld a,(basepos):inc a:inc a:ld (missile),a   ;X co-ord to approx
         ld a,23:ld (missile+1),a                    ;that of the Base.
         ret


 CONTROL_MISSILE
         ld a,(missile):cp 0:ret z                   ;If no missile, return
         ld ix,space:call PRINT_MISSILE              ;Erase the missile
         ld a,(missile+1):dec a:ld (missile+1),a     ;Move it up
         cp 0:jr z,stopmissile                       ;Check for scren top
          ld ix,missilegr:call PRINT_MISSILE          ;Print missile
          ret
  stopmissile                                         ;Missile has reached
          ld a,0:ld (missile),a                       ;top, so reset it.
          ret


  PRINT_MISSILE
          ld a,(missile)
          cp 0:ret z
          ld hl,(missile):jp PRINT_CHAR
```

The two routines INVADERS and BASE are called repeatedly until a change takes place in the variable 'flag'. When things are progressing normally, flag contains the value zero. When things are not progressing so well, flag is given a special value:

Value of flag	Meaning
0	Everything OK
1	User press ESCAPE key
2	The aliens have landed
3	The aliens bombed the player successfully
4	The player has destroyed all the aliens

Each of these values is checked for in turn and the flow of control is re-routed to the appropriate subroutine to take care of things. If flag now contains three, the number of lives the player has is reduced by one. If that means the player has no lives remaining, the program will start again from the beginning and print the title page, because the game is over.

This brings us to the next section which sets up all the variables. There is nothing much to be said of this part, as it involves nothing more than poking a number of addresses with starting values. SET INVADERS uses a small loop to set all 28 elements to one. This is the array which determines whether a particular alien is alive or dead.

The next section controls most of the writing to the screen. TITLE SCREEN is my favourite piece of code. It puts my name on the screen and waits for the player to hit the spacebar. You can put your name here but you will have to wait until next month to do so, as both the strings of data to be printed and the printing routine are held over until then.

The procedure SET SCREEN will select Mode 1 and draw all the shields and scores using routines such as PRINT SCORE. PRINT SCORE, in common with the other number-printing routines, uses the routine PRINTHL which is defined next month. We have already looked at this routine in some depth.

The final section is a rather complicated one which prints and moves the player's base along the bottom of the screen. Also here is the code which checks to see if a missile has been fired and, if so, prints and moves it.

• Next month I will complete the listing.