PLAYER MISSILE EDITOR

Artworx

Artworx Software Company
150 North Main Street • Fairport, New York 14450

PLAYER MISSILE EDITOR

written by Dennis Zander

This program is designed to facilitate the creation of players and missiles for the player missile graphics system of the Atari computer (see ref. 1 & 2). A player is a column one byte wide which more than covers the screen vertically. Up to four players may be "drawn", each on its own one byte by twenty byte area. The players are drawn one point at a time, producing up to four figures in graphics mode 0. The program calculates the equivalent byte values for the eight points in each row of a player, shows the figure in real size and stores the data in strings as part of the program. Groups of figures (or players) may be viewed quickly in real size or tested in the demo game. The demo can be set to use each of four players for movement in a different direction, or to sequence through the four players for animation (framing) effects. Players can then be moved around the screen in the demo/game mode by using a joystick.

The game Demo shows how fast players may be moved on the screen using strings in BASIC. This technique has been described by George Blank in Creative Computing (see ref. 3). An outline of the PMEDIT program is given in Appendix 1.

PLAYER/MISSILE WITH STRINGS

This whole system of moving players or rapidly changing a particular player is based upon the use of machine language routines that are part of the BASIC cartridge. In Atari BASIC a string is nothing more than a sequence of bytes; one byte per character in the string. Any value from 0 to 255 is legal although '34' and '155' are not usable because of the way in which data is stored. If we could get BASIC to think that P1$ is player 1 then when we say P1$(Y)=SHIP$: P1$(Y+1)=SHIP$: P1$(Y+2)=SHIP$ etc., the string SHIP$ and the equivalent player generated will very rapidly be moved down the screen in the direction of increasing Y values (see Lines 50-75 in Appendix 3 for example). The entire string is moved regardless of size, as long as no errors are created (i.e. P1$ is too short). The player SHIP$ must start and stop with a blank or you will leave a trace behind as you move up and down. This is the reason for the INSERT 1 option of the editor just in case you forget to leave a blank at the top.

To rapidly change players for animation, explosions, etc., you simply equate strings:

```
P1$(Y) = WALKER1$
P1$(Y) = WALKER2$
P1$(Y) = WALKER3$
P1$(Y) = WALKER4$
```

The demo game shows this with the ALTERNATE function. Lines 380-398 of the program show an effective way in which to cycle player shapes. Substituting each of four or more players for one displayed player in a cycle as the player is moved across the screen can produce a very animated player. Try the robot provided in group 3 in Alternate mode!

It is quite easy to move horizontally; this is done with a simple POKE 53248, X. It is a good idea to equate PLX to 53248 and thus use "POKE PLX,X". The whole player, a 128 byte or 256 byte (double or single line resolution) column is instantly moved to the new X position. The horizontal position registers, as they are called, are at:

53248 PLAYER 0
53249 PLAYER 1
53250 PLAYER 2
53251 PLAYER 3

POKE PLX +2, X will move player 2 to the location X. While players are a full byte wide, all four missiles are stored in the same column of bytes, two bits per missile. Missiles are not automatically fired, but must be moved just like a player. One difficulty with this method is that the four missiles are all part of the same byte so that moving vertically with the string method may cause some missiles to be erased when passing each other vertically. Each missile has its own horizontal register, 53252 - 53255 are missiles 0 - 3 respectively. The memory requirements for players and missiles are summarized in Appendix 2.

Fooling BASIC into storing the strings in the player missile area when running a program is done in lines 9100 - 9155 of the PMEDIT program.

9115 Checks for the end of memory and steps back enough pages to allow for the graphics mode which you are using.

9120 Gets the address of the variable table.

9125 Gets the address of the string storage area.

9130 Calculates the new offsets from where BASIC thinks the string is to where we want it to be.

9140 - 9155 The offsets, array size and number of elements stored are poked into the appropriate addresses.

IMPORTANT! This method of using strings for players assumes that the first variable defined in your program is as shown in Line 10, even if you have a U=1 ahead of this it will not work! If you add Line 10 to an existing program, it will not work because other variables were historically defined first! However, all is not lost. By LISTing your game program to cassette or disk after adding Line 10 as the first variable and typing NEW and ENTER "C:" or ENTER "D:GAMENAME", BASIC will reorder the variables as though everything were typed in fresh. On a frequently edited program, this also can reduce the memory required.

EDITOR

The editor may be entered after selecting player resolution, by typing an 'E' when the program comes up or by pressing START when in the demo mode. When first entering the editor, you will be viewing player Group 0. As supplied, Group 0 contains only one player 8 bytes in length. The following discussion will explain the options used in the editor. It is followed by a sample practice session.

OPTIONS

Options are selected by typing the first letter of the option when the first letter is highlighted (B - bytes) or the labeled key if the word is highlighted (DELETE key for example).

B - BYTE sets the number of bytes used, 1 - 20. T2o digits must be entered even if the number of bytes is less than ten (i.e. for 8 bytes enter 08) but no 'RETURN' is required. If you mistype, enter the second digit and type 'B' again.

P - PLAYER sets the number of players in this player group (1 - 3). Note that the players are also designated by directions: up, dn, rt, le; for up, down, right and left respectively. When in demo mode, the player specified will be used for the direction indicated. If there are less than four players, then player 1 will also be player 3 and player 0 will also be player 2. Player 0 will also be player 1 if there is only one player.

D - DEMO takes you to the game demo. The START button returns you to the editor.

C - COLOR lets you set the color for each player of the group being edited. Left - Right on the joystick changes the hue and Up - Down changes the luminosity. Pushing the trigger saves the player color. All colors are stored when the group is STORE'd.

G - GROUP selects the player group to be edited and sets the identifying number which it will be stored as (two digits must always be used as in BYTE). If the group number is 6 for example, then the player strings will be stored in lines 20060 through 20065 (LINE = 20000 + 10 * 6). The group specified by the number does not come up until the EDIT key is pressed.

J - JOIN/SEP puts the real size players in the lower left corner either edge to edge or separates them. The key is alternate acting.

E - EDIT displays the player group specified in GROUP for editing purposes.

S - STORE saves the players appearing on the screen into strings in the lines indicated by the GROUP number. At this time you will also be asked for a group name which will be put into a remark at the end of the player string. IMPORTANT! Stored players are only saved as part of the program in RAM and do not become permanent until you type 'Q' for QUIT which will then save everything onto disk or cassette.

Q - QUIT saves the entire program with any new data strings to disk or cassette. The program tests to see if there is a disk present, and if so will then automatically save the file PMEDIT. See lines 700 - 750 of the program code.

N - NEXT will call up the next player group for editing if pressed once. By pressing 'N' and holding it, the program will do a quick scan through the groups showing only the real size players in the lower right. The new group will come up blank and then Group 0 will be shown again.

R - REVERSE will reverse the dark and light areas of each player displayed (i.e., make a negative image).

U - UNDO will restore all players displayed to the form currently stored.

CLEAR - Pressing the CLEAR key will erase all displayed players.

INSERT - Pressing the INSERT key once will insert one blank line at the top of each player (in case you forget to put one in). A blank line is required at the top and bottom of each player to keep them from leaving trails on the screen when moved.

DELETE - The DELETE key will permit you to clear out stored player groups. All groups from the number typed to the end will be deleted. This is useful for starting fresh, but be careful, a RETURN with no number will abort with no deletions. One group should always be left.

EXAMPLE EDITING SESSION

1) Use DOS to make a duplicate working disk before starting this example session, or CLOAD and then CSAVE onto a blank cassette.

2) CLOAD the cassette or boot the disk.

3) Type 'E' to enter the editor after selecting line resolution.

4) Type a 'G', then '03' and 'E' to edit. The four views of the robot should be displayed on the screen.

5) Type 'R' for reverse. Type 'J' and the four players in the corner will be separated. Type 'J' again and they will be joined.

6) Type 'P', Type '2'

7) Type 'B', Type '16'.

8) Press the INSERT key.

9) Press 'CLEAR' to clear the display.

10) Type 'U' and everything is back to normal!

11) Type 'C' and use the joystick to set the color. Press the trigger to save the color and return to editing.

12) Use the joystick to move the cursor around and the trigger to plot or delete points. The cursor wraps around end-to-end, top-to-bottom.

13) When you are satisfied with your new creation, Type 'G', Type '04'. This will save it as group 4 (current new group) when you type 'S'.

14) Type 'S' - Type (a four character name) push RETURN and watch it be entered as lines 200040 - 20044.

15) If you want to see it move, continue or else go to Step 22.

16) Type 'D' - go to Demo (see Demo Game description).

17) Type '04' RETURN, Type 'D' RETURN for directional substitution of player.

18) Now you can use the joystick to move the player(s) around.

19) The SELECT key will let you load a different group or the same one with a re-selection of ALTERNATE/DIRECTIONAL mode.

20) The START key will take you back to the editor.

21) Push START and get back to it!

22) You should be in the editor with the last Group you worked on showing and NEW GROUP = 5.

23) Type 'G', Type '00' and 'E'. You should now be back to the flying saucer.

24) Type 'N' - a spaceship!

25) Push 'N' down and hold it. Each of the player groups will be shown in rapid sequence in the lower right corner. Releasing the 'N' will stop at the next group.

26) Now you can either delete the group you created by pushing DELETE and typing '04' or Type 'Q' and this will save the entire program and any new groups you may have created onto cassette or disk.

DEMO GAME (LINES 300 - 499)

The Demo Game is an illustration of how the players look when you move them. It also can be played as a game. You start in the center of the screen and try to get to the sides as many times as possible. Running into a colored block on the top or bottom will blow you up and put you back at the center. Whenever you run into something, it's playing field number is shown. This gives you some idea of how the collision detection works.

You can change groups by pressing the SELECT key which will also allow you to change player modes from Directional to Alternate.

Directional means player 0 is used for UP, player 1 is used for RIGHT, player 2 is used for DOWN and player 3 is used for LEFT.

Alternate means that the players 0 - 3 will alternately be substituted for player 0, which is a fine way to see the effects of animation.

By pressing START you can return to the editor.

GAME CONSTRUCTION

The listing in Appendix 3 shows the lines required to use strings with player - missile graphics. Lines 50 - 150 show a fast way to move a player with the GOSUB call of Line 1000. Lines 1000 - 1070 move the player and fire a "drawn" laser. Line 20001 contains the data for the player.

Line 10, Lines 9005 - 9017 and 9100 - 9240 are the only lines required as the basis for using this method. These may be LISTed from the PMEDIT program to cassette to create a list file. The list file can be ENTERed to form the basis for a game. If you have the disk version, then the program in Appendix 3 is included as PMEX file.

Certain changes are required depending upon the resolution and graphics mode desired.

In Line 9115, the 'E' must be replaced with a number appropriate for the graphics mode used.

GR 0	'E' (8)
GR 1	8
GR 2	8
GR 3	8
GR 4	8
GR 5	16
GR 6	16
GR 7	24
GR 8	36

To change to single line resolution, change Line 9115, SZ=256:SH=1:SL=0 and Line 9210, POKE 559,62.

REFERENCES:

1. "Player-Missile Graphics with the Atari Personal Computer System", Chris Crawford, COMPUTE, January 1981, pp. 66 - 72.
2. "Outpost: Atari", George Blank, CREATIVE COMPUTING, January 1981 (Player Missile Graphics).
3. "Outpost: Atari", George Blank, CREATIVE COMPUTING, April 1981, pp. 194 - 198.

APPENDIX 1

PROGRAM DESCRIPTION (see program listing)

1	- 11	HEADER
200	- 204	DEMO/EDITOR SELECT
300	- 499	GAME
300	- 330	INITIALIZE
335	- 360	DRAW PLAYFIELD
365	- 398	MOVE PLAYER
400	- 440	ADD OBSTACLES
445	- 458	CHECK PLAYER-PLAYFIELD COLLISION
460	- 499	SELECT OPTIONS
500	- 8840	EDITOR
500	- 640	SELECT FUNCTION
700	- 750	SAVE PROGRAM DATA
1000	- 1090	STORE PLAYER DATA AS STRING
1100	- 1180	EDIT PLAYER
2000	- 2060	FLASH CURSOR DURING EDIT
3000	- 3070	LOAD PLAYER TO SCREEN
4000	- 4040	REVERSE LIGHT/DARK AREAS
5500	- 5550	CALCULATE BYTE FOR STRING FROM SCREEN DATA
6000	- 6295	UNDO: REESTABLISH ORIGINAL PLAYER
6400	- 6420	JOIN PLAYERS - SEPARATE PLAYERS
6500	- 6590	INSERT 1 LINE AT TOP OF PLAYERS
7000	- 7050	CLEAR SCREEN OF PLAYER
7400	- 7440	SET PLAYER COLOR
8000	- 8150	SET NUMBER OF BYTES USED
8200	- 8300	SET NUMBER OF PLAYERS USED
8400	- 8460	SET GROUP NUMBER USED
8500	- 8540	DELETE LINE NUMBERS FOR ONE PLAYER
8800	- 8840	DELETE PLAYERS STARTING AT...
9000	- 9088	INITIALIZATION - TITLE
9100	- 9155	SET-UP PM AREA AS STRINGS
9200	- 9240	CLEAR PLAYER MISSILE STRINGS
9300	- 9399	TITLE PAGE & "CLIMBER"
20000	-	PLAYER DATA STORED AS STRINGS

APPENDIX 2

USEFUL ADDRESSES (all values in decimal)

559 put a 62 here for single line, a 46 for double line resolution

623 sets player/playfield priorities (one bit on)
    1: all players have priority over all playfield registers
    4: all playfield registers have priority over all players
    2: mixed. P0 & P1, then all playfield, then P2 & P3
    8: mixed. PF0 & PF1, then all players, then PF2 & PF3

704 color of player-missile 0
705 color of player-missile 1
706 color of player-missile 2
707 color of player-missile 3
53248 horizontal position of player 0
53249 horizontal position of player 1
53250 horizontal position of player 2
53251 horizontal position of player 3
53252 horizontal position of missile 0
53253 horizontal position of missile 1
53254 horizontal position of missile 2
53255 horizontal position of missile 3
53256 size of player 0 (0=normal, 1=double,3=quadruple)
53257 size of player 1
53258 size of player 2
53259 size of player 3
53277 A 3 here enables player-missile graphics, a 0 disables them
54279 put high byte of PMBASE here

OFFSETS

PMBASE	DBL-LINE	SNG-LINE
M$(miss.)	+384	+768
P0$(plyr0)	+512	+1024
P1$(plyr1)	+640	+1280
P2$(plyr2)	+768	+1536
P3$(plyr3)	+896	+1796

APPENDIX 3

```
0 REM PLYR/MISS EXAMPLE
1 REM MOVE PLYR W/JOYSTICK IN POS.1-USE TRIGGER TO FIRE!
2 REM by DENNIS R. ZANDER
3 REM (C) COPYRIGHT 1981
10 DIM M$(1),P0$(1),P1$(1),P2$(1),P3$(1),P$(80),C$(20):REM THESE MUST BE THE FIRST VARIABLES ENTERED!
20 GRAPHICS 7+16:X=120:Y=70:D=0.7
30 GOSUB 9005:GOSUB 9200:GOSUB 20000:GOSUB 70:GOTO 1000
50 XD=1:YD=1:Y=Y+1:GOTO 75
60 XD=D:YD=-D:Y=Y-1:GOTO 75
70 XD=1:YD=0
75 X=X+1:POKE PLX,X:P0$(Y)=PL0$:RETURN
90 XD=-D:YD=D:Y=Y+1:GOTO 115
100 XD=-D:YD=-D:Y=Y-1:GOTO 115
110 XD=-1:YD=0
115 X=X-1:POKE PLX,X:P0$(Y)=PL0$:RETURN
130 XD=0:YD=1:Y=Y+1:P0$(Y)=PL0$:RETURN
140 XD=0:YD=-1:Y=Y-1:P0$(Y)=PL0$:RETURN
150 RETURN
1000 TRAP 1000:GOSUB STICK(0)*10
1010 IF STRIG(0) THEN 1000
1020 XP=X-45:YP=Y-13
1030 COLOR 3:PLOT XP,YP:TRAP 1040:DRAWTO XP+30*XD,YP+30*YD
1035 FOR I=110 TO 114:SOUND 0,I,8,8:NEXT I:SOUND 0,0,0,0
1040 COLOR 0:PLOT XP,YP:TRAP 1000:DRAWTO XP+30*XD,YP+30*YD
1050 GOSUB STICK(0)*10
1060 IF  NOT STRIG(0) THEN 1050
1070 GOTO 1000
9005 LINE=20000:IX=0:IY=1:CH=764:PLX=53248:X0=175:Y0=91:CONS=53279:CPF=53252:O=0:U=1:T=2:E=8
9010 DIM CX$(U),D$(E),K$(U),L$(3),BT$(3),NM$(4):CX$=" "
9015 DIM BL$(20),BLST$(20),PL0$(20),PL1$(20),PL2$(20),PL3$(20),PLYR$(20)
9016 BLST$="6z":REM SEE LINE 9016 OF PMEDIT
9017 BL$="":REM SEE LINE 9017 OF PMEDIT
9100 REM ***SETUP PM AREA AS STRINGS**
9101 REM 9100-9235 SAME AS PMEDIT EXCEPT FOR CONSTANTS.
9115 PMA=PEEK(106)-24:SZ=128:SH=0:SL=128
9120 A=PEEK(134)+PEEK(135)*256
9125 ST=PEEK(140)+PEEK(141)*256
9130 FOR I=0 TO 4:OFFS=PMA*256+SZ*3+SZ*I:HI=INT((OFFS-ST)/256)
9140 POKE A+I*E+T,OFFS-ST-HI*256:POKE A+I*E+3,HI
9145 POKE A+I*E+4,SL:POKE A+I*E+5,SH
9150 POKE A+I*E+6,SL:POKE A+I*E+7,SH
9155 NEXT I:RETURN
9200 REM ***CLEAR PLYR/MISS AREA******
9210 POKE 559,46:POKE 54279,PMA:IF SZ>128 THEN POKE 559,62
9220 POKE 53277,3:CP0=120:P0$(1)="":P0$(SZ)="":P0$(2)=P0$:P1$(1)="":P1$(SZ)="":P1$(2)=P1$
9227 P2$(1)="":P2$(SZ)="":P2$(2)=P2$:P3$(1)="":P3$(SZ)="":P3$(2)=P3$
9230 FOR I=0 TO 3:POKE PLX+I,X0+I*8:NEXT I:POKE 704,CP0-80:POKE 705,CP0-80:POKE 706,CP0:POKE 707,CP0
9235 P$(80)="":P$(1)="":P$(2)=P$:RETURN
10000 REM YOUR OWN PLAYER CAN BE PUT AT LINE 20000
20000 PL0$="$<":REM SEE LINE 2000 OF PMEDIT
20001 POKE PLX,X:P0$(Y)=PL0$:RETURN
```