TRICKY TUTORIAL ™

#4

BASICS OF ANIMATION

BY ROBIN ALAN SHERER

TRICKY TUTORIAL #4

BASICS OF ANIMATION

by
Robin Sherer

HOW TO LOAD

TAPE...

Place the tape in your recorder, label side up. Make sure the tape is rewound, and the BASIC Cartridge is in place. Also, reset the counter to zero. Push PLAY on the recorder and type RUN"C: and press RETURN. If the program won't start to load, try positioning forward or backward a little. The easiest way to find the beginning is to listen to the "noise" on the tape with a regular recorder. When you find the steady tone, you have the beginning of the program. We recommend you write down the number on your recorder's counter as each program example starts. This will make it easier to find each part later on.

DISK...

To load and run the disk, first turn on your disk drive. When the busy light goes out, place the disk in the drive. Now turn on the computer, with the BASIC Cartridge in place. The program will load each part and run by itself.

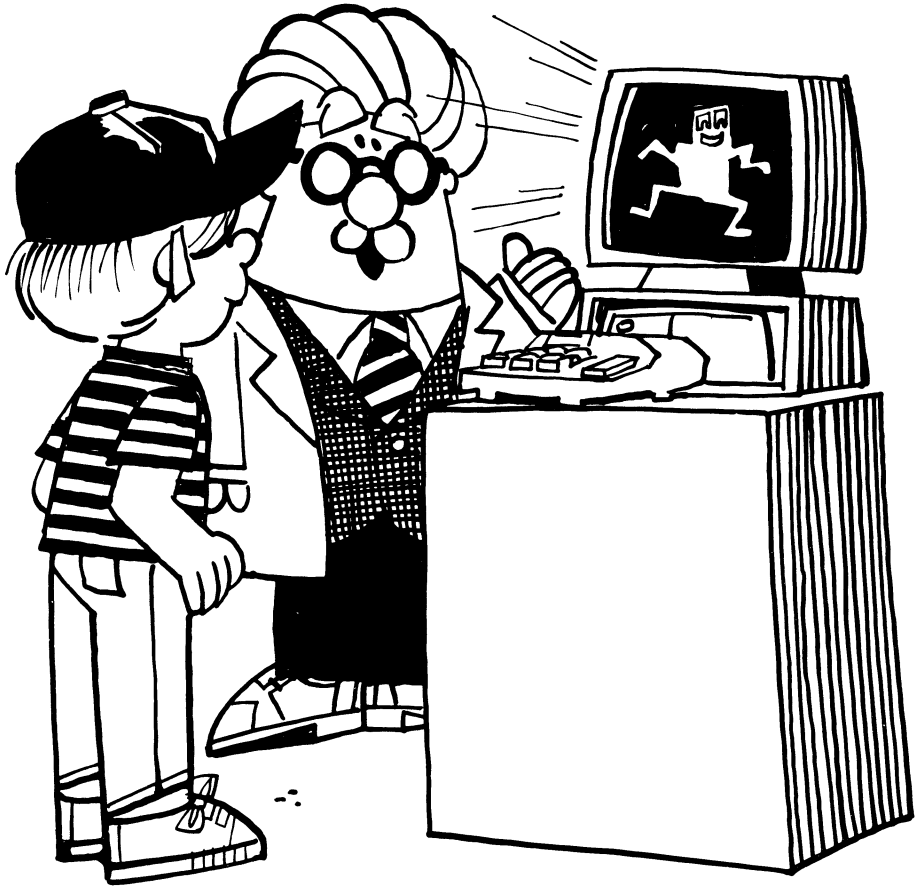
Any defective tapes or disks should be returned to:

Educational Software Inc.
4565 Cherryvale
Soquel, CA 95073
(408) 476-4901

LET THE ANIMATION BEGIN!

What does animation mean to you? Were you hoping that by purchasing this program you would then be able to do cartoons on the screen in intricate detail...or did you perhaps just hope to learn some basics to move a few shapes around the screen for a game or business application you are programming? Well, we're going to start out this lesson by discussing what can and also cannot be done on your ATARI.

Animation requires two main qualities to appear nicely on your screen. First, you must control enough points so that something seems to be happening. Then you must move these points around fast enough so that they appear life-like.



Rule number one:
IT CAN'T BE DONE FROM BASIC

Wait! Don't panic and return this program to us before you read on. We know you are not a Assembly language programmer. Neither are we, except when necessary.

BASIC in the ATARI is slow because it is an interpreter, meaning that every time you tell it to go move some point around on the screen, it has to first go and figure out exactly how to do what you've asked. This takes so much time that even with the machine doing hundreds of thousands of steps per SECOND, it can only move a single point around as fast as our first two examples. The way around this is to do one of four things:

1) You can program in the machine's language, where it doesn't have to interpret. This, however, would be too difficult for the average ATARI user.

2) You can use machine language routines built into the machine. A simple example of this is what we are doing in Part 2 using the PRINT command to draw some what complicated shapes very quickly. In fact if you look at the code for the space bug, you will see a delay loop was put in to slow it down! Also, although too complicated to explain in a introductory program like this, you can redefine the characters you are printing to draw almost anything you can imagine. The monsters in SPACE INVADERS are redefined letters of the alphabet, and are moved about by PRINT commands.

3) You could also go buy a larger BASIC that includes some machine language movement routines to move your shapes around. (Basic A+ or Microsoft Basic).

4) You can use PLAYER/MISSILE GRAPHICS like the tie fighter program we include as an example.

ANIMATION USING THE PLOT COMMAND

If you haven't already done so, now is the time to load in the program. Follow the instructions in the program and use your Joystick to move a simple square around the screen. This is similiar to the movement in games of the SURROUND type. As you can see while you follow the program , it is really quite simple to move a square! We show you the code here, so you can see the necessary steps to move the square. Don't worry about learning it all now.

THIS IS THE CODE FOR MOVING A SQUARE:

```
10100 GRAPHICS 4:POKE 764,255:X=20:Y=12
10103 ? "HERE IS OUR SQUARE":? "USE YOUR JOYS
TICK TO MOVE IT ....WHEN DONE PRESS ANY KEY"
10105 V=STICK(0)
10110 Y=Y+(V=13)+(V=9)+(V=5)-(V=10)-(V=14)-(V
=6)
10120 X=X-(V=10)-(V=11)-(V=9)+(V=6)+(V=7)+(V=
5)
10145 IF X<1 THEN X=1
10146 IF X>79 THEN X=79
10147 IF Y<1 THEN Y=1
10148 IF Y>39 THEN Y=39
10150 COLOR 1:PLOT X,Y:IF PEEK(764)=255 THEN
10105
```

The better games of this type add another player to compete with and leave a controlled trail behind them. The trail is kept track of by the computer. Then, when you touch a location where the computer's "records" show a trail is, it scores for the other player

After you have run the program through to the point where we show you how to erase the trail, we suggest you stop the program (press BREAK and remove your tape or disk). You can always come back and finish it later. To continue with disk, type RUN "D:PLOT2.DSK". For tape, use RUN"C:". The reason for stopping now is for you to write a small program that makes use of a square being moved using the PLOT statement. The easiest way would be for you to look at the listing.

HERE'S THE CODE FOR MOVING A SQUARE, BUT WITHOUT A TRAIL:

```
10520 V=STICK(0)
10525 SETCOLOR 2,0,0:COLOR 2
10530 IF V=14 THEN Y=Y-1:PLOT X,Y+1
10540 IF V=13 THEN Y=Y+1:PLOT X,Y-1
10550 IF V=11 THEN X=X-1:PLOT X+1,Y
10560 IF V=7 THEN X=X+1:PLOT X-1,Y
10575 IF X<1 THEN X=1
10586 IF X>79 THEN X=79
10590 IF Y<1 THEN Y=1
10600 IF Y>39 THEN Y=39
10610 COLOR 1:PLOT X,Y:IF PEEK(764)=255 THEN
10520
```

Lines numbered from 10520 to 10610 are the ones that move the square around. Add to our program using your own ideas. Also, since the code exists in the program we sent you, please feel free to delete the lines you don't need, and start with them already in the computer to save time (the Assembler cartridge could do this quickly!). Please be sure if you save your work, it is on a separate tape or disk.

NOTE: Even if you intend to do only non-game applications, the learning you gain from writing a simple game will help other attempts at animation.

Before going on, we want to point out that when using PLOT and erasing your "trail", you have to actually PLOT the position you are erasing with the color of the background. This makes that position seem to disappear as you PLOT the next position in the color of your choice.

The next thing I do to show you the BASICS OF ANIMATION is to draw out the famous "Tie Fighter" shape and move it using the PLOT command.

PLEASE NOTE THAT THIS IS ONLY AN EXAMPLE AND THE SHAPE COULD BE ANYTHING FROM A BUSINESS TO A HOME APPLICATION. IT ALSO COULD BE MUCH LARGER THAN WE DREW. I keep our programs simple to allow everyone to understand the PRINCIPLES INVOLVED, but with more memory and time you can do great things like this:

Here is how to input points to plot:

1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1
1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2
1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3
1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4
1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5
1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6
1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7
1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8

Figure 1.

Look at Figure 1. I drew out a shape at the upper left corner of what is called an X-Y coordinate system. The upper left most point that is filled in is at x=1 and y=1. Moving down the figure the next point is at x=1,y=2, then x=1,y=3, etc. with the last point of OUR shape at x=5,y=5. The computer doesn't care what order you input points into PLOT statements, just input them all. This means you don't have to use the points in the same order as we did, or even use the same points. Now, to put these into a BASIC program we use the DATA statement (read your BASIC manual!). It looks as follows:

```
DATA 1,1,1,2,1,3,1,4,1,5,2,3,3,2,
3,3,3,4,4,3,5,1,5,2,5,3,5,4,5,5
```

Every two numbers represent one point to plot.

Here is the complete TIE FIGHTER code:

```
11820 GRAPHICS 4:X=40:Y=20:COLOR 1:GOSUB 1193
0:? "USE JOYSTICK TO MOVE FIGHTER":? "PRESS A
NY KEY TO GO ON"
11825 POKE 764,255:SPEED=5:SETCOLOR 2,0,0:TRA
P 11820:S=SPEED
11830 V=STICK(0):IF V=15 THEN 11832
11831 GOTO 11835
11832 IF PEEK(764)=255 THEN 11830
11833 IF CNTR=2 THEN 12600
11834 GOTO 12000
11835 COLOR 2:GOSUB 11930
11841 Y=Y+(V=13)*S+(V=9)*S+(V=5)*S-(V=10)*S-(
V=14)*S-(V=6)*S
11842 X=X-(V=10)*S-(V=11)*S-(V=9)*S+(V=6)*S+(
V=7)*S+(V=5)*S
11920 COLOR 1:GOSUB 11930:IF PEEK(764)=255 TH
EN 11830
11930 RESTORE 11960:POKE 752,1
11940 READ A,B:IF A=0 THEN 11970
11950 PLOT A+X,B+Y:GOTO 11940
11960 DATA 1,1,1,2,1,3,1,4,1,5,2,3,3,2,3,3,3,
4,4,2,4,3,4,4,5,3,6,1,6,2,6,3,6,4,6,5,0,0,0
11970 RETURN
```

The last part of this lesson allows you to move the cursor up to the data line we used for the tie fighter shape. The program is now stopped, waiting for you to enter new numbers into the DATA statements. If a mistake occurs, you can try pressing RESET and typing RUN, but if the DATA Statements get too messed up it may be easier to reload in the program. You probally won't have to go through this since it is easy to change the required lines in the program.

All you need to do is type in new numbers that you have chosen, based on a drawing of YOUR shape made on standard grid paper. Once the DATA statement looks correct, press RETURN to enter it and then type CONT and press RETURN again to restart the program. Also, you may use numbers bigger than the 8,8 for your shape, but when you move it, if it goes off screen it will bomb the program since we only allowed in our code for the smaller shape. The more points you use, the slower it goes. Again, feel free to copy and modify our program or, if you like, just slowly change a small part of it to see how your own changes can add to it or make it better!!!

ANIMATION USING THE PRINT COMMAND

We included the "Bird at the Ocean" from the ATARI BASIC manual as a convenience for those who never typed it in. It demonstrates the simple use of alternating between two shapes in a PRINT command in order to obtain a feeling of motion. We will use this method now. Again, after finishing this part please modify the shapes or the background to get a feel for your own ideas. Backgrounds are simply created with PLOT & DRAWTO commands and the special Graphics Characters. We next show you animation of a single character in the horizontal and vertical direction. They both have simple sounds to demonstrate how easy sound is to add.

HERE'S THE CODE FOR THE SPACE BUG:

```
21310 GRAPHICS 18:? #6;"NOW LETS TRY A SPACE
BUG":FOR W=1 TO 600:NEXT W
21320 TRAP 23512
21330 ? "K":X=1:Y=10:POKE 752,1:CNT=0
21340 POSITION X,Y
21345 ? "  ":POSITION X,Y+1
21350 ? "◥◤":POSITION X,Y+2
21360 ? "▟▙":POSITION X,Y+3
21370 ? "▌ ▐"
21391 SOUND 1,210,10,CNT+2
21392 FOR W=1 TO 20:NEXT W:POSITION X,Y
21393 ? "  ":POSITION X,Y+1
21394 ? "  ":POSITION X,Y+2
21395 ? "  ":POSITION X,Y+3
21396 ? "  "
21397 X=X+2:POSITION X,Y
21400 ? "  ":POSITION X,Y+1
21410 ? "◥◤":POSITION X,Y+2
21420 ? "▟▙":POSITION X,Y+3
21430 ? "\\"
21431 FOR W=1 TO 20:NEXT W:POSITION X,Y
21432 ? "  ":POSITION X,Y+1
21433 ? "  ":POSITION X,Y+2
21434 ? "  ":POSITION X,Y+3
21435 ? "  "
21440 SOUND 1,160,10,CNT+4:X=X+2:IF X>37 THEN
21460
21450 GOTO 21340
21460 X=1:CNT=CNT+1:IF CNT=5 THEN 21480
21470 GOTO 21340
21480 SOUND 1,0,0,0:IF COUNT=1 THEN 22560
```

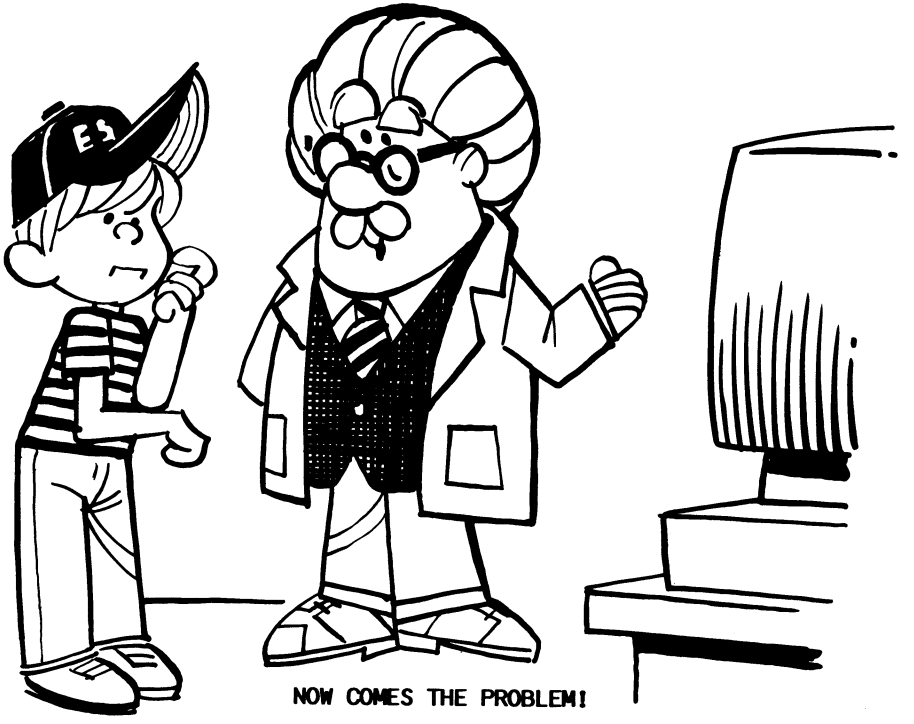
The program will stop and allow you to move the cursor up to the six lines that hold the shape for the "SPACE BUG". Change (using the special graphic characters) the shapes ONLY BETWEEN the quotes. When done, move the cursor to the top line and press RETURN until all six lines are re-entered into memory. If you goof up, clear the page and type GOTO 23500 and press RETURN. If you enter the new shape correctly move to a BLANK area on the screen and type "CONT", then press RETURN to see your shape move across the screen. When you have played enough with our code, try adding backgrounds first before your shape is printed or perhaps try several shapes. You will quickly notice a main drawback to animation using the PRINT command (and PLOT too); everything you move over is rewritten and thus disappears

-Note-

To do full animation many additional tricks are taught in our other Tutorials. This is not intended just to sell more programs, but you deserve to know how to use the full power of your ATARI. Some of these other Tutorials include PAGE FLIPPING, MODIFICATIONS TO THE DISPLAY LIST AND SCROLLING.

PLAYER MISSILE GRAPHICS

You probally already have heard of the ATARI's unique feature called PLAYER MISSILE GRAPHICS (PMG). What PMG does is allow you to animate simple shapes around on the screen without having to redraw them as we did in the earlier lessons using PRINT and PLOT. This capability is built into the hardware, so all we have to do is program the hardware with simple POKE commands.



NOW COMES THE PROBLEM!

What makes PMG hard to explain is that there are so many built in features, and so many POKEs to do. We wanted to reach a compromise with an introductory lesson like this, so we include an example using PMG with some explaination. Like before, if you will just take the time to modify our program you will see the effects of each change. It would be even better if you had a specific goal in mind, say to change the shapes of the Players, or a different background. Please note that the way this example was written is not the best it could have been done. It was writtrn long ago, but it works and that was the goal. The various POKEs are listed in more detail in our MASTER MEMORY MAP, and PMG is more fully explained in a seperate Tutorial(#5).

HERE'S THE PGM CODE:

```
20 K=15:REM INITIAL POS OF P/M 1
30 TRAP 20
40 SOUND 3,13,8,2
50 PRINT "{CLEAR}"
60 GOSUB 680:REM BACKGROUND SUBROUTINE
70 SETCOLOR 2,0,0:X=120:Y=90:REM SET BACKGROU
ND COLOR AND PLAYER POSITION
80 POKE 704,133:REM SETCOLOR PLAYER0 TO BLUE
90 POKE 705,198:REM SETCOLOR PLAYER1 TO GREEN
100 A=PEEK(106)-8:POKE 54279,A:PMBASE=256*A:R
EM SET PLAYER-MISSLE STARTING ADDRESS
110 POKE 559,46:POKE 53277,3:REM ENABLE PM GR
APHICS WITH 2-LINE RESOLUTION
120 POKE 752,1:REM MAKE CURSOR INVISABLE
130 FOR I=PMBASE+512 TO PMBASE+640:POKE I,0:N
EXT I:REM CLEAR OUT PLAYER0 FIRST
140 FOR I=PMBASE+640 TO PMBASE+768:POKE I,0:N
EXT I:REM CLEAR OUT PLAYER1 FIRST.THIS IS TO
PREVENT RANDOM JUNK.
150 FOR I=PMBASE+640+K TO PMBASE+644+K:READ B
:POKE I,B:NEXT I:REM DRAW PLAYER1
160 DATA 153,189,255,189,153
170 REM DATA FOR TIE SHAPE
180 FOR I=PMBASE+512+Y TO PMBASE+516+Y:READ A
:POKE I,A:NEXT I:REM DRAW PLAYER0
190 POKE 53256,0:POKE 53260,18:REM SIZE OF PL
AYER0 AND ALL MISSLES
200 POKE 53257,0:REM SIZE OF PLAYER1
210 DATA 153,189,255,189,153
220 POKE 656,3:? "PRESS 1 TO STOP GAME"
230 REM SAME SHAPE FOR OTHER PLAYER
240 REM NOW COMES THE MOTION/COLLISION ROUTIN
ES
250 IF PEEK(764)=31 THEN 1650
260 F=PTRIG(0):REM READ TRIGGERS
270 G=PTRIG(1)
280 IF F=0 OR MIS=1 THEN GOSUB 390:REM SLOW M
ISSLE ROUTINE
290 IF G=0 THEN GOSUB 520:REM FAST MISSLE MOV
E ROUTINE
300 POKE 656,0:IF FS=1 OR GS=1 THEN ? "GREENS
  SCORE=";SCOREG;"          BLUES SCORE=";SCOREB
310 FS=0:GS=0:REM THESE TELL ATARI TO PRINT S
CORE
320 POKE 53278,0:REM CLEAR OUT COLLISION REGI
STERS TO USE AGAIN
330 A=PADDLE(0)
340 B=PADDLE(1)
350 POKE 53248,A:IF MIS=0 AND A<>0 THEN POKE
53252,A-1:REM MOVE PLAYER0(AND MISSLE0) TO N
EW LOCATION INSTANTLY!
```

```
360 POKE 53249,B:POKE 53253,B-1:REM SAME FOR
PLAYER1
370 GOTO 250:REM GO READ PADDLES AGAIN
380 REM SOUBROUTINE FOR BLUE  MISSILE
390 SOUND 0,227-V*2,8,INT(16-V/10):REM MAKE S
OUND DECREASE AS MISSILE MOVES
400 V=V+1:POKE PMBASE+384+Y-V,0:POKE PMBASE+3
83+Y-V,1
410 E=PEEK(53256):IF E>0 THEN HIT=1:REM TEST
FOR HIT!
420 IF V>95 THEN 500:REM MISSILE MOVED FAR EN
OUGH
430 MIS=1:RETURN
440 HIT=0:F5=1:REM SAYS TO PRINT NEW SCORE,MI
S & HIT ARE COUNTERS
450 SOUND 1,221,10,12:POKE 53249,250:FOR W=1
TO 200:NEXT W
460 SOUND 1,0,0,0
470 POKE 53249,120:POKE 53278,0:REM REPOSITIO
N PLAYER1 AND CLEAR COLLISION REGISTER
480 SCOREB=SCOREB+1
490 V=0:MIS=0:SOUND 0,0,0,0:RETURN
500 IF HIT=1 THEN 440
510 V=0:MIS=0:SOUND 0,0,0,0:RETURN
520 SOUND 0,200,8,8:P=0
530 FOR I=15 TO 130:POKE PMBASE+383+I,0:POKE
PMBASE+384+I,4:  P=PEEK(53257)
540 IF P=3 THEN 590
550 NEXT I
560 SOUND 0,0,0,0
570 RETURN
580 REM P=PEEK(53257):IF P=2 THEN RETURN
590 SOUND 0,0,0,0:G5=1:REM SAYS TO PRINT NEW
SCORE
600 RESTORE 630:SOUND 1,121,8,12
610 FOR I=PMBASE+512 TO PMBASE+640:POKE I,0:N
EXT I:REM CLEAR  OUT PLAYER FIRST
620 Y=75:FOR I=PMBASE+512+K+Y TO PMBASE+516+K
+Y:READ B:POKE I,B:NEXT I:REM DRAW PLAYER
630 DATA 153,189,255,189,153
640 FOR I=384 TO 512:POKE PMBASE+I,0:NEXT I
650 SOUND 1,0,0,0
660 SCOREG=SCOREG+1
670 RETURN
680 GRAPHICS 8:CNT=0
690 COLOR 1:X=RND(0)*319:Y=RND(0)*159:CNT=CNT
+1:PLOT X,Y:IF CNT>150 THEN RETURN
700 GO TO 690
```

Player Missile Graphics Explanation

LINE 50 : Clear the screen

LINE 70 : The X and Y position of player one is established here.

LINES 80-90 : Player colors are stored in 704 to 707.

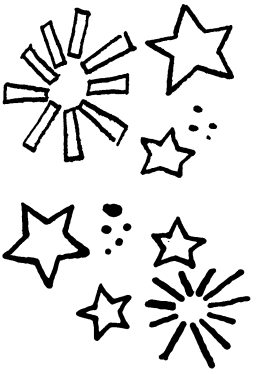
LINE 100 : Memory location 106 holds the value (in # of PAGES) of your top of memory. We simply subtract 8 pages to make room for the shapes of the Player to be stored.

LINES 130-140 : To create or erase a player, you POKE the shape into memory where we reserved it in line 100. We first erase the player by Poking in all 0's. Player 0 starts at the value of PMBASE plus 512 and goes for 128 locations in memory. Player 1 starts at 640.

LINES 150-160 : Read the shape into memory.

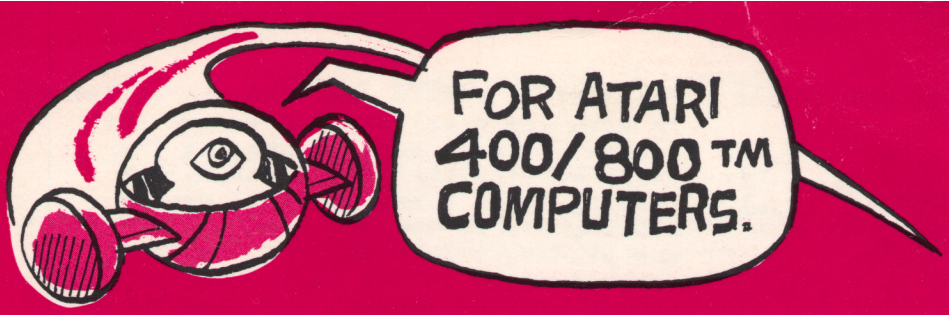
LINES 250-370 : Main loop of program. Location 764 holds the last key pressed. When you run the program, you will note that the missiles don't move the same. One moves in a loop that is fast because it doesn't go back and allow any players to move. The other is slow because it does. The collision registers mentioned in the comments to line 320 "record" when certain things touch eachotheron the screen. You determine what this location looks for by the value you put into 623 (if desired). Lines 350 & 360 POKE the location that controls where ACROSS the screen the Players will appear. If you don't plug in Paddles the values transfered here will be 0, so the players will be off screen. Values of about 40 to 200 will be on the screen.

LINES 390-finish : the rest is really just two subroutines to move the missiles up and down the screen. This is done by erasing the shape at it's current location, and drawing it at a new place in memory placing 0's at the old memory locations, and placing 1's in memory). If you draw it close to the last location, the motion will be slower, but smooth. We also add to the scores here when the collision registers at 53256/7 don't hold a 0. Finally, the routine at line 690 just creates stars by random PLOTs on the screen.



The only way that all of these POKEs will become familiar to you is to get a hold of one of the publications that gives detailed descriptions of each memory location needed for PMG. Our Master Memory Map is good, and ATARI's Operating System manual is better (but very hard to read). We hope you feel that this lesson was worth the cost.

Thanks.....bye!



BASICS OF ANIMATION

Basics of Animation consists of a set of simple programs designed to teach those new to computers how to make shapes appear to move around on the screen. The three methods demonstrated are animation using the PRINT COMMAND, PLOT COMMAND, and ATARI'S great PLAYER/MISSILE GRAPHICS.

The person using this lesson should be familiar with BASIC programming so that he or she can read the code that is included. Since the program is not protected, the user is encouraged to try their own modifications inorder to gain a greater perspective of the art of animation.

This program requires 16k of memory for TAPE users and 24k for those using DISK.