XY TUTORIAL

By: Timothy Hays©

A Tutorial on the BALLY BASIC XY Command
For Exceptionally Well Controlled Graphics

Introduction: PART 1

Turn on the ARCADE unit and your TV set and load the first program on the "XY TUTORIAL" cassette. This program will introduce you to the theory and technicalities of the XY command. It incorporates the XY command as it's main function. As soon as the program loads, IMMEDIATELY stop the tape recorder. It is a very short program. Watch it loop a few times. What it is doing is choosing a random value for XY (XY=RND (22526)-11263) and plotting it. Don't worry about these numbers yet though, but you might want to know that the number given to choose a value from (22526) covers the entire screen. In theory, this program chooses a random location on the screen and draws a line from there to the center, refer to FIGURE 1 (next page). In actuality, this is not what is happening. You may notice some lines forming above and below the center of the screen (0,0) which are not aimed at the center (Fig. 2). This is what appears to be happening; when the random location is chosen, sometimes it might be located to the right, or left, OFF the screen (Fig. 3). If you are familar with the &(9) function, you know that there are many screen 'windows' to the right and left of your main view (Fig. 4). The ARCADE only lets the graphics stay within the 'window' currently on the screen, so the lines that come from somewhere off the screen (another 'window') are kept in your screen view in the manner shown in Figure 5. When a line goes off the screen on the right side, it will appear on the left side and continue at the same angle it was originally following. In other words, if a line (coming from the center) went off the screen at 79,20 it will appear at -79,20 (opposite side) and continue. Refer to line 'A' in Fig. 5. The lines going

Listing of program #1

```
    1 .TIM HAYS
    4 .RND XY VALUE TO PER
SPECTIVE POINT (Ø,Ø)
    5 CLEAR ;NT=Ø
    7 BC=RND (256);FC=BC+4
+RND (32)×8
    8 FOR A=1TO 1ØØ
   1Ø XY=RND (22526)-11263
   2Ø LINE Ø,Ø,1
   3Ø NEXT A;RUN
```

FIGURES 1-6....

FIG. 1

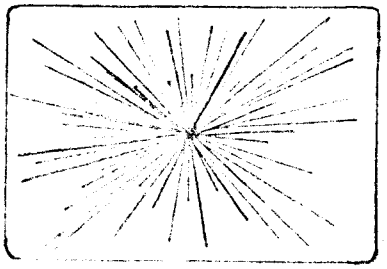
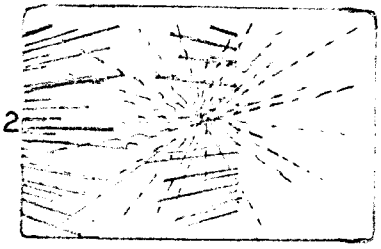


FIG. 2



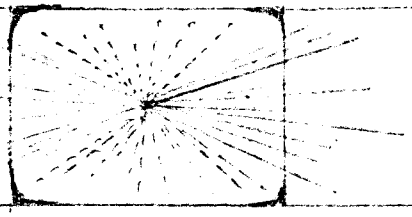


FIG. 3

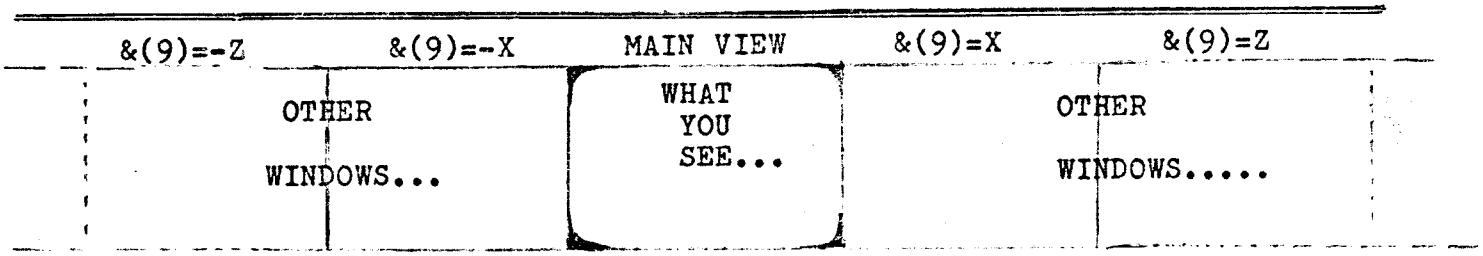


FIG. 4

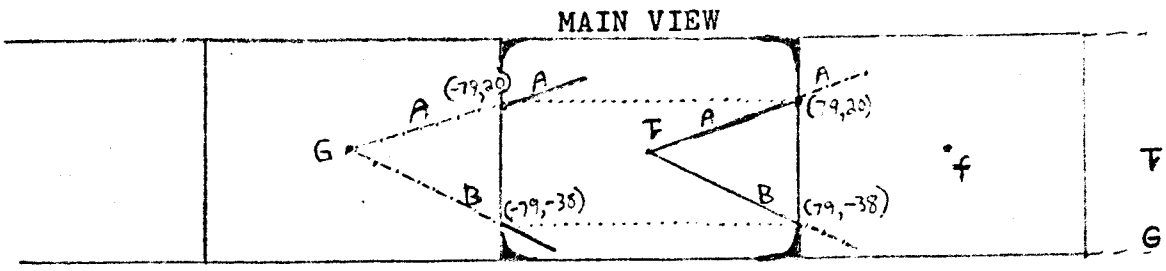


FIG. 5

F-True point (Ø,Ø) of MAIN VIEW

G-True point (Ø,Ø) off left side of screen(MAIN VIEW)

e-Imaginary point (Ø,Ø)off right side of screen

f-True point (Ø,Ø) off right side of screen

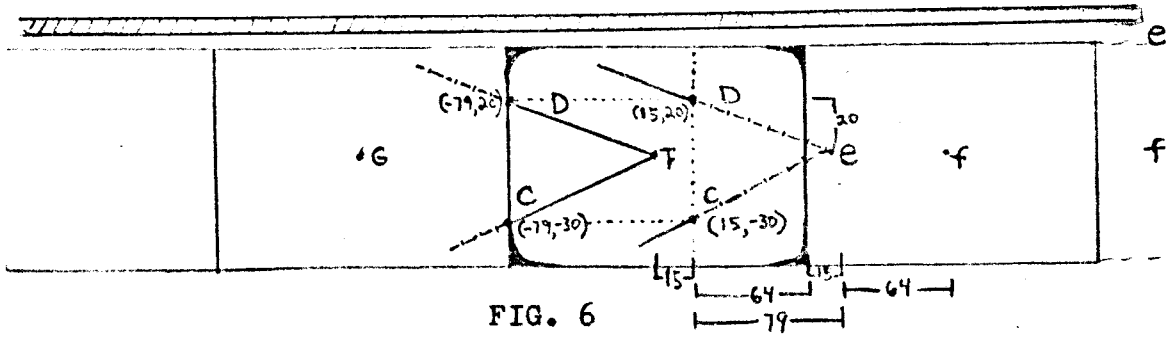


FIG. 6

off the on the left side are not following the same exact rule, they appear on the right side alright, but they start drawing at X=15. (Fig. 6) Don't worry about this though, it isn't very important, it's just to let you understand what is happening.

The actual program is contained in lines 8, 1Ø, 2Ø, 3Ø. Line #7 adds color. You may want to see what this does with reverse lines, insert 2Ø LINE Ø,Ø,3. About this time some of you may think that the same effect can be achieved without the XY function. True, it can, but one line of:' 1Ø XY=RND (22526)-11263 ' is a lot less memory consuming and takes much less time than:

```
1Ø A=RND (159)-79
2Ø B=RND (87)-43
3Ø LINE A,B,2        ;.
4Ø LINE Ø,Ø,1
```

.Performs the same oper-
.ation, but w/no efficiency
invisible

The object of this tutorial is to eliminate the use of these invisible lines (3Ø LINE A,B,2). Invisible lines cause everything to be done in a roundabout way. XY statements are a very direct and comparatively fast way to control graphics.

PART 2

Another VERY useful idea in the controlling of the XY command is the incorporation of loops. The next program will show how these become so greatly important. RESET the computer and load the next program (Demonstration program #2). This program shows some video art that depends on the XY function for it's speed. It may remind you of the BALLY spokes program, but it has no relation to it as you can see by studying the program. Now watch the program loop a few times, notice the speed in which the graphics are executed in spite of the decoding the machine has to do for each new XY coordinate. Now halt the program. Do NOT reset! Clear the screen. Press-- :INPUT ,and 'GO' and turn on the

PART 2 cont....

tape recorder to load some changes to the same program. What you see printing out on the screen is correct. Those statements on the screen are altering the statements currently in the program. Now watch the program run again. The graphic designs are completely different, the XY locations are the same, but the spots where the lines were drawn to (pinpointed) were simply changed to get a completely different effect. Color was also enhanced with the additions. (subroutine at lines 7ØØ, 71Ø, 73Ø) You may have noticed what appeared to be curves in 2 of the loops. These types of routines can be very useful in applications where you would want to plot a curve, or just to show 3-D simulation. There is plenty of memory left in this program, but if I needed to, I could cut down the memory requirements considerably. I kept the format in an easy-to-figure-out style. How to do loops like these, and figuring out the needed XY values for the loops will be explained next. Listing of this program supplied.

PART 3

This next program should be invaluable to your understanding of the XY command. Reset, and load the next program. It is very short and to the point. Input a number on the X plane (-79 to +79), then input a number on the Y plane (+44 to -44) & press 'GO'. Then the computer prints the value of XY at that given coordinate, and draws a line from that point to the center of the screen (Ø,Ø). This line is to confirm that the coordinate you gave is the one you meant. To do another plot, just press any key. Example: say that you want to know the XY value for the coordinate 48,-13. When the screen says: INPUT X , you press 48 on the keypad & 'GO'. The screen says: INPUT X 48
INPUT Y you press -13 and

T. H.

LISTING OF PROGRAM #2

```
 1 NT=Ø;.DEMONSTRATION PROGRAM #2
 2 .TIM HAYS ,XY STATEMENTS
 3 BC=RND (256);FC=BC+4+RND (32)×8
 5 B=RND (13)+1
 8 C=B×256;CLEAR
1Ø FOR A=-1Ø831TO 11185STEP C
2Ø XY=A
3Ø LINE Ø,-43,1
4Ø NEXT A
5Ø FOR A=11185TO 11343STEP B
6Ø XY=A
7Ø LINE Ø,-43,1
8Ø NEXT A
9Ø FOR A=11087TO -10929STEP -C
1ØØ XY=A
11Ø LINE Ø,-43,1
12Ø NEXT A
13Ø FOR A=1TO 7ØØ;NEXT A
14Ø BC=RND (256);FC=BC+4+RND (32)×8
15Ø CLEAR
2ØØ FOR A=11185TO 11343STEP B
21Ø XY=A
22Ø LINE -79,-43,1
23Ø NEXT A
24Ø FOR A=11185TO 11343STEP B
25Ø XY=A
26Ø LINE 79,-43,1
27Ø NEXT A
28Ø FOR A=-10831TO -10673STEP B
29Ø XY=A
3ØØ LINE Ø,43,1
31Ø NEXT A
32Ø FOR A=1TO 7ØØ;NEXT A
33Ø BC=RND (256);FC=BC+4+RND (32)×8
335 CLEAR
34Ø FOR A=11185TO 11343STEP B
35Ø XY=A
36Ø LINE -79,-43,1
37Ø NEXT A
38Ø FOR A=11087TO -10929STEP-C
39Ø XY=A
4ØØ LINE -79,-43,1
41Ø NEXT A
42Ø FOR A=-10929TO -11087STEP -B
43Ø XY=A
44Ø LINE 79,43,1
45Ø NEXT A
46Ø FOR A=-10831TO 11185STEP C
47Ø XY=A
48Ø LINE 79,43,1
49Ø NEXT A
5ØØ FOR A=1TO 7ØØ;NEXT A
51Ø RUN
6ØØ CLEAR ;PRINT
61Ø PRINT " TURN OFF TAPE
RECORDER!
62Ø PRINT " PRESS ANY KEY
63Ø A=KP;RUN
```

T.H.

ADDITIONS TO PROGRAM #2,
PART 2....

```
 4 &(9)=2Ø
 9 Y=Ø;GOSUB 7ØØ
3Ø Y=Y-1;LINE Ø,Y,1
45 Y=Ø;GOSUB 7ØØ
7Ø Y=Y-1;LINE Ø,Y,1
85 Y=Ø;GOSUB 7ØØ
11Ø Y=Y-1;LINE Ø,Y,1
15Ø CLEAR ;GOSUB 7ØØ
22Ø LINE -79,43,1
23Ø NEXT A;GOSUB 7ØØ
26Ø LINE 79,-43,1
27Ø NEXT A;GOSUB=7ØØ
3ØØ LINE Ø,Ø,1
31Ø NEXT A;GOSUB 7ØØ
335 CLEAR ;GOSUB 7ØØ
36Ø LINE Ø,15,1
37Ø NEXT A;GOSUB 7ØØ
4ØØ LINE Ø,15,1
41Ø NEXT A;GOSUB 7ØØ
44Ø LINE Ø,-15,1
45Ø NEXT A;GOSUB 7ØØ
48Ø LINE Ø,-15,1
49Ø NEXT A;GOSUB 7ØØ
```

SUBROUTINE... CHANGE COLOR
SCHEME. PART 2

```
7ØØ FC=RND (256);A=FCXRND
(8);&(2)=A;&(3)=A
71Ø &(Ø)=BC;&(1)=BC
72Ø RETURN
```

'GO'. The computer now prints out the XY value (XY= -3280) and draws a line from the designated spot (48, -13) to the center (Ø,Ø). Here is a listing of the program:...

```
  1 .
  2 .CARTESIAN COORDINATES TO XY VALUES ROUTINE
 10 NT=1;CLEAR ;PRINT
 20 INPUT " INPUT X"X
 30 INPUT " INPUT Y"Y
100 A=Y×256
105 IF X<ØA=A+256
110 XY=A+X
120 PRINT " XY=",XY
130 LINE Ø,Ø,1
140 D=KP
150 RUN
```

Lines 100, 105, 110, are what does the actual figuring and converting, don't be concerned with how this works now, it will be explained later. Lines 140 & 150 are to run the program again when you press any key. This program should be very handy when you have a list of coordinates (13,-45 -56,-27 etc.) and you need to know the XY equivalents of them. The program is short enough to just key-in by hand whenever you need it. Now halt the program and enter this line: &(1Ø)=250 no line number is needed. This will open up the screen 'curtain' all the way and show the Bally's scratchpad twinkling at the bottom. Now run the program. This next item might pleasently suprise you; try entering some Y values below the bottom of the screen. Example: enter 20 for X and -60 for Y. The line goes beyond the -43 (bottom of the screen) limit! When the screen curtain is at the normal setting (&(1Ø)=174) the line graphics can only be drawn down to Y=-43, the limit of the screen, but by raising the screen curtain all the way, graphics can be drawn down to Y=-60 !! You can experiment using different numbers for X, and keeping Y at -60 by first entering Y=-60, and then deleting line 30. When you experiment with graphics going into the scratchpad area, some wierd things begin to happen. Colors change, music sounds, and the print statements are even changed to

PART 3 cont....

form some type of cypher. Some of these cyphers are fun to figure out, experiment!! What appears to be happening is, the lines that go into the scratchpad area are actually altering the memory and making some wierd things happen. To really see some neat things happen, enter this next program by hand and run it. (remember to reset first!)....

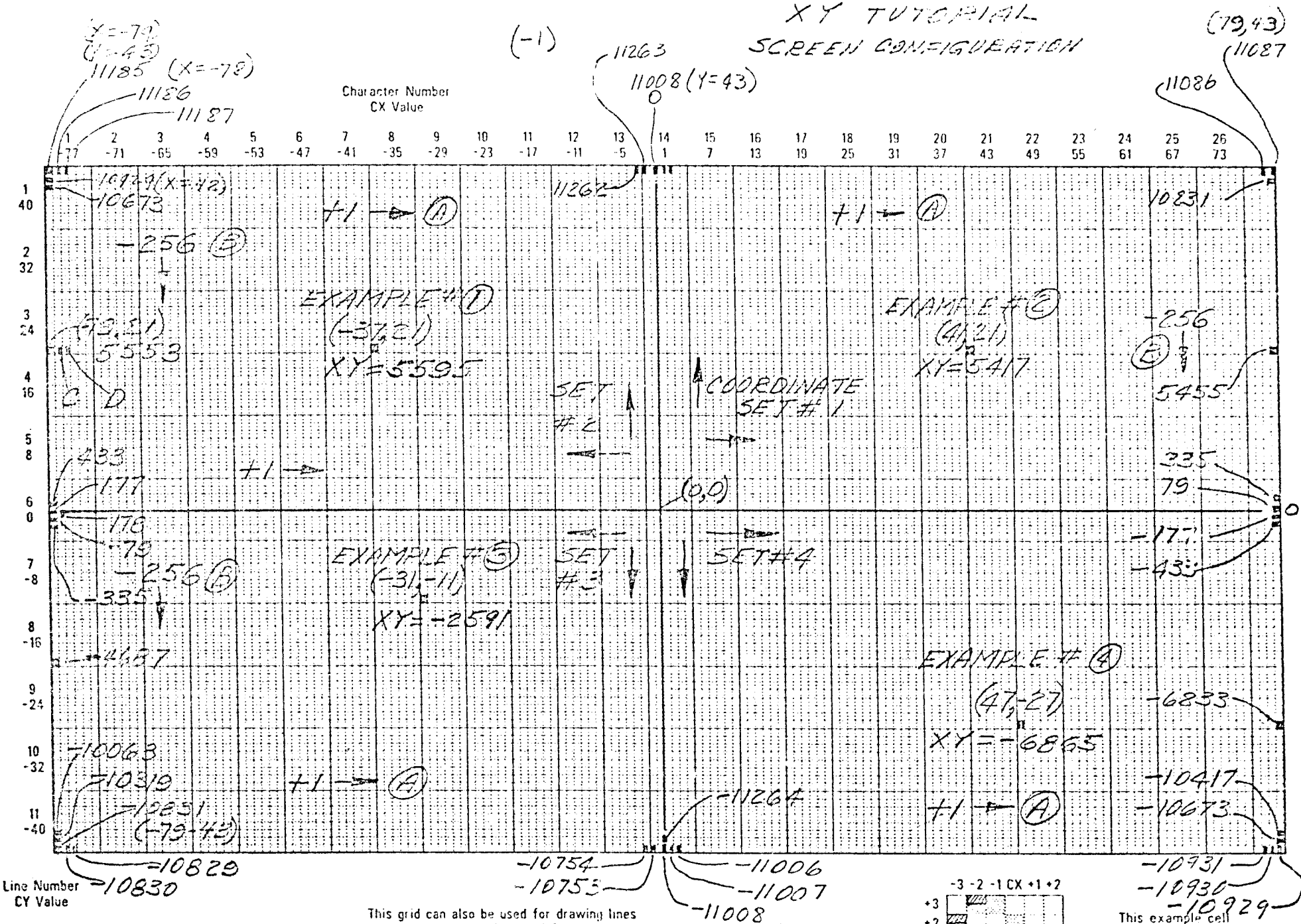
```
A=700                       ;.no line #, try different -
10 A=A+3 .(or 1,2,etc.)     ;.values for A (up to 32767)
20 XY=%(A)                  ;.peeks from memory
30 LINE Ø,Ø,3
40 RUN
CLEAR;RUN
```

This program peeks the XY value from the memory, in doing so, it seems to alter memory the same way that the last program did. It may lock-up (crash) a few times, but it is a very fun program to watch and listen to. If it does crash, just re-inter it, it's very short. Enter a new value for A, set &(10)=255, clear the screen and run. Try A=2ØØØ. After a short time, the lines begin erasing themselves.

PART 4 VARIABLES

In order to do loops, you need to know your needed sets of variables or points to draw the lines from. In this section, refer to the next page which is a graph/chart called 'SCREEN CONFIGURATION' of all the important number values needed to find a value of any given point on the chart. The most often used starting points in video art or other graphic routines are the 4 corners (top-right, top-left, bottom-left, bottom-right). XY equivalent values of those corners are written on the chart. The second most often used points are, the center of any edge of the screen (79,Ø; -79,Ø ; Ø,43 ; Ø,-43). The very center of the screen (Ø,Ø) is XY=Ø, this is used often both as a starting

XY TUTORIAL
SCREEN CONFIGURATION



This grid can also be used for drawing lines by using the "coarse" CX and CY values then adding or subtracting the needed amount to position the line exactly as you want it. Use the example cell to the right.

EACH PIXEL TAKES UP 2 BITS OF SCREEN MEMORY

This example cell is shown 2.5 times scale size.

Notice that no pixels are used along the right and bottom border.

#8

point, and an end point. When you are programming a game, and need a set of XY values, you either use the "CARTESIAN COORDINATES TO XY VALUES ROUTINE" (program in PART 3) and figure them out beforehand, or you can use this chart and the method given here to quickly find your needed sets of values. We shall now use example #1 on the chart. Say, you want to draw a line from (-37,21) to the center of the screen (Ø,Ø). Find your location (pixel) on the chart. If you look directly to the left, (direction of the arrow) on the edge of the chart, there is a number : 5553. This means that the dot (pixel) on the edge of the chart(-79,21), the XY value equals 5553. The pixel directly to the right of it equals 5554 (letter 'C'). The pixel to the right of 'C' equals 5555 (letter 'D') and so on. Values of all pixels increment by +1 to the right of any other pixel (direction of 'A'). A pixel directly to the left of a given pixel,'X', is equal to X-1. Look at the top left corner of the chart, that pixel equals 11185 (X=-79, Y=43). The pixel to the right of that equals 11186, next one to the right equals 11187, then 11188, 11189, and so on. This continues until you meet (-1,43), where the values change (one pixel to the left of top-center). Using example #1, you use the number on the left edge of the chart (5553) which is at X=-79, the point you want is at X=-37. Subtract -37 from -79, the answer is 42, add 42 to 5553 and you get 5595 which your needed value; (-37,21)=5595 in XY terms. Example #2, the value you need is at (41,21). On the right edge of the chart, on the same 'Y' plane (Y=21) is the XY value 5455, (at point 79,21). Subtract 41 from 79 (X values for both points), the answer is 38. You are moving to the left now, so you subtract 38 from 5455, the answer is 5417, your needed value. But what if you don't have a number to refer to on the same 'Y' plane? Example

PART 4 cont....

#3 shows how this can be calculated. The value you need is at point (-31,-11). If you look all the way to the left side of the chart, on that same line (Y=-11), there is no reference number, nothing. Once you get familar with the configuration of all the numbers on the chart, the method used in examples #1 & #2 will become simple. There is an alternate method to be used only if you have an 'ARCADE' unit with you. Whenever you clear the screen, XY=Ø. First, clear the screen, then enter: LINE -31,-11,1;PRINT XY which draws a line from the center to point (-31,-11) and prints out your needed XY value. You may think this method is simpler, it is, but it's a hassle if you happen to be programming at the time, or if you are creating a program without using the computer (such as creating a flowchart). We will now get back to the first method. In examples #1 & #2 you only had to add or subtract by 1 to move your variable until you were at the correct spot, this was by moving only left & right. You now need to know how to move your variables up and down the chart. By studying the chart, you may see that if you take any given point on the screen, the pixel direc'ly above that is 256 more. Look to the center-left of the chart. The pixel there equals +177, if you add 256 you get +433 which is the next pixel above it. When going down the chart, simply subtract 256 for each pixel downward. In example #3, the closest known pixel on the left edge equals -4687 which is at X=-79, example #3 is at X=-31.(-79 minus -31 equals -48.) Subtract -48 from -4687, the answer is -4639. There are 8 pixels seperating the two points vertically. 256 times 8 equals 2048. Add 2048 to -4639, the answer is -2591, now look on the chart at exam-

T.H.

XY TUTORIAL

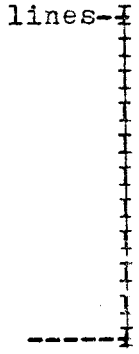
Listing of last program

3-D Forward Simulation Above A Flat Plane

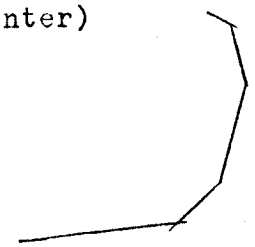
```
 1.
 2.3-D FORWARD SIMULATION ABOVE A FLAT PLANE
 4.TIM HAYS (c)
 5.9/3TO 1Ø/8/79
 1Ø &(9)=175
 2Ø CLEAR ;P=Ø
1ØØ FOR V=ØTO -6STEP -1
11Ø T=-Vx256
18Ø IF JX(1)CLEAR
19Ø X=X+JX(1)x2;W=Xx256
2ØØ XY=177-W;LINE 79,X,1
21Ø XY=433-W-T;LINE 79,X+W+1,2
22Ø XY=-79-W-T;LINE 79,X+V-1,1
225 IF V<-2XY=-335-W;LINE 79,X-3,1
23Ø XY=-177-W-T;LINE 79,X+V,2
24Ø XY=-591-W-T;LINE 79,X+V-3,1
25Ø XY=-335-W-T;LINE 79,X+V-2,2
26Ø XY=-1615-W-T;LINE 79,X+V-7,1
27Ø XY=-1359-W-T;LINE 79,X+V-6,2
28Ø XY=-3663-W-T;LINE 79,X+V-15,1
29Ø XY=-3407-W-T;LINE 79,X+V-14,2
3ØØ XY=-7759-W-T;LINE 79,X+V-31,1
31Ø XY=-7503-W-T;LINE 79,X+V-3Ø,2
4ØØ XY=-2639-Wx3;LINE -18,-2,1
41Ø XY=-4687-Wx3;LINE -15,-3,1
42Ø XY=-7759-Wx3;LINE -9,-2,1
43Ø XY=-10817+Xx8;LINE -5,-1,1
44Ø XY=-1Ø777+Xx8;LINE -1,-1,1
45Ø XY=-10983+Xx8;LINE 1,-1,1
46Ø XY=-1Ø943+Xx8;LINE 3,-1,1
47Ø XY=-7857+Wx3;LINE 9,-2,1
48Ø XY=-4785+Wx3;LINE 15,-2,1
49Ø XY=-2737+Wx3;LINE 18,-2,1
5ØØ P=P+1;R=P+3;S=P+5
51Ø XY=475-P-W;LINE -12-R,6+R,1
52Ø LINE Ø,S,1
53Ø XY=292+P+W;LINE 12+R,6+R,1
54Ø LINE Ø,S,1
55Ø IF P>9ØRUN
6ØØ NEXT V
61Ø GOTO 1ØØ

PRINT SZ
    957
```

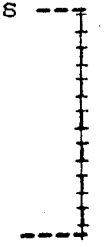
2ØØ through 31Ø for horizontal lines--



4ØØ through 490 for vertical lines (converging at center)



mountains --



memory left is more than half

pushing joystick to the left or right moves the view as would a joystick in an airplane

T.H.

PART 4 cont....

ple #3 and see that it equals -2591. I know this may seem complicated at first but you'll get used to it. Now see if you can figure out how to do example #4, all the variables are given including the answer for comparison to what you get.

PART 5 - TRUE 3-D SIMULATION

You now come to the 6th and last program, RESET & load it into the computer. This program simulates a perspective (3-dimensional) view of a flat surface which has been plotted out on a perspective grid, and a view of mountains in the distance gradually getting closer. It would simulate the forward view from an airplanes cockpit as it gradually closes up on the mountains. Just watch it for a while and note the speed at which all lines are executed. By giving XY coordinates you don't have to draw invisible lines to get a solid line to start where you want it to. This saves a lot of processing time and saves an enormous amount of memory. SZ=957, more than half the memory is left! Just think what could be done if all programs could do this much graphically and still have more than 1K of memory to do whatever you want! Of course, this program could still be cut down considerably, I left it this way so you won't have more trouble trying to figure out what I did. All variables were figured out beforehand as described in PART 4. Now find your hand-controler, push the joystick to the left or right. Everything moves as if you had the stick in the plane! This is a very pleasant sight, but you cannot move too far to either edge. When doing this, the program might jam-up, the reason for this was explained in PART 3. Notice that everything stays in full proportion while the view banks from side to side. Even the mountains bank to the side. No special formula was used, everything was accomplished with a calculator and the 'Screen Configuration' chart. Repeating loops are very important in programs like these. You should now look at the listing of Program #2 and take single loops out of it and get familar with what each one does, start with lines 1Ø through 4Ø.

T.H.