GRAFIX SAMPLER

An Intro to Commodore 64 Graphics

- Arcade critters, custom text fonts
- Electronic schematic, music notation
- Point and line plotting
- 3-D object rotation
- Sprite graphics
- Over 20K of fun!

MIDWEST
MICRO associates

'64 GRAFIX SAMPLER

for the COMMODORE 64
by Dr. Jim Rothwell

CONTENTS:

These routines are offered for your enjoyment and as a point of departure for exploring the exciting graphics potential of your '64. Most examples are written in BASIC for easy examination and interaction. Also, a machine language routine is included to better display the true speed of the '64 and to provide skilled programmers with an application tool they can extract.

Best of luck with your '64 !

1. Character Demo

This demo contains a variety of techniques which display the 64's ability to use alternate character sets. The bit-pattern for these characters is stored in a table beginning at memory location 8192. A technique was used to chain together the character table as part of the BASIC program for loading directly into memory. Line 1ØØ POKES locations 45 and 46 to restore the actual end of the BASIC program; POKEs to 56 and 52 protect the table.

Note: The subroutine at 42Ø switches to the alternate character set. The subroutine at 48Ø switches to the standard set.

Several display techniques are used in this demo to give you an idea of the wide range of possibilities. "Arcade Characters" are POKEd to the screen, with a corresponding POKE made to the Color Table for each character. This straightforward method is also used for "Custom Character Sets". "Hi-Rez Grafix" employs PRINT statements, with each statement utilizing the specific locations in the character set which are defined as electronic symbols. Note that it takes several character locations combined to display one part, such as a transistor or resistor. Truly modular components! "Music Notation" uses PRINT statements again, but this time the characters are defined as string variables. See if you can tell which pieces are which notes!

2. Point and Line Plotting

Each routine is identified in the LISTing. Special mention should be made of the machine language program which is POKEd to memory [lines 134Ø to 143Ø]. This clears the Hi-Rez screen and also sets it up for a specific color. This M/L routine is used in lines 127Ø to 129Ø. Note: the screen color is POKEd into location 2 before the routine is called. See what happens with various values.

Incidentally, the program will execute faster without all the Remarks and extra line numbers with colors. These help the for-matting of the LISTing but also slow down the execution a bit (or is it byte?). You may remove them if you wish.

3. 3-D Demo

This demo consists of 4 essential routines which are used by each of the 3-D demos. These common parts are:

1)SCREEN CLR and SETUP: lines 1230-1250

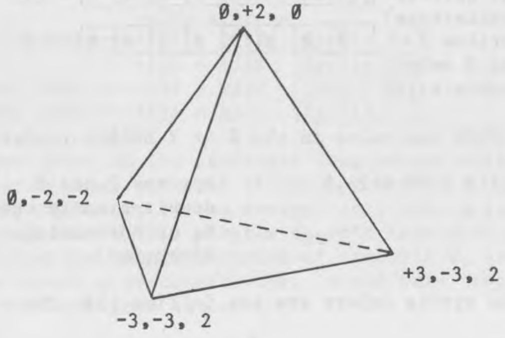
2)POINT PLOT: lines 1180-1200; this routine is also called by Line Plot.

3)LINE PLOT: lines 1140-1170 plot all the points between the end-points.

4)3-D ROTATION: lines 1000-1110. Reads a set of three points (X,Y,Z-axis) and transforms them according to these variables: XR=X rotation; YR=Y rotation; ZR=Z rotation.

The X-axis is viewed as running left-and-right (as in a normal graph); the Y-axis is up and down; and the Z-axis represents near and far (or the axis perpendicular to the face of the screen).

For simplicity, 30 different positions of rotation are allowed (12 degrees each) around each axis. Thus, the variables XR, YR, and ZR can range from 0 to 29. For each example, the X,Y,and Z points are stored on separate lines. The X-points are read first, then the Y-points, and finally the Z-points. For convenience, each object is defined symmetrically around an arbitrary center (0,0,0), so that points to the left of center, below center, or "behind" center are negative, while points to the right of center, above center, or in "front" of center are positive. Points for each set are listed in order:(X,Y,Z).



EXAMPLE

Note that as each view is drawn, it is placed to the right of the previous display. The routines have a built-in offset which places each consecutive example to the right of the previous one.

4. Sprite Demo

Sprites are much like portable drawing pads which can be designed, colored, and moved around the screen in a variety of ways. This demo guides you through a simple implementation of sprite graphics in BASIC. REMarks are abundant for identifying essential parameters.

The various sprite registers are set in lines 4Ø-14Ø: X is the horizontal position on the screen; Y is the vertical position on the screen.

The PLANE is represented by Sprite Øa (with prop) and Øb (without prop). The two versions are alternated to give the propeller an appearance of motion. Sprite 1 is the CLOUD; Sprite 2 is the BOAT.

The plane and cloud are shown in double-size mode. Experiment with the double-size mode by changing the POKE values in lines 11Ø and 12Ø. To compute the POKE value, total the corresponding bit values for each sprite you wish to make double-size using the byte diagram below. (Zero bits will result in normal-size sprites.)

Sprite	7	6	5	4	3	2	1	Ø	TOTAL
(all sprites double-size)	128	64	32	16	8	4	2	1	= 255
(sprites 2 and Ø only double-size)	Ø	Ø	Ø	Ø	Ø	4	Ø	1	= 5

Then POKE the value in the X or Y EXPAND register:

```
110 POKE R+23,5          (Sprites 2 and Ø
                          double-size in the
                          X, or horizontal,
                          direction)
```

The sprite colors are set in line 13Ø. The value

POKEd is one less than the number shown on the color key. For example, RED is key number 3, so the POKE value for RED is 3-1, or 2. Change the plane to red by: POKE C1,2

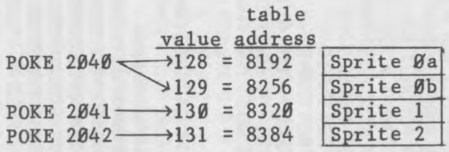
The data is tranferred from the data statements to the sprite tables in line 15Ø. Lines 2ØØ-25Ø print the sea and island. Their colors are determined by control characters embedded at the beginning of the print statements in line 19Ø and 21Ø. Use the CTRL+color key, while in the quote mode, to change these colors.

All three sprites are enabled by poking the variable EN with the value of 7, computed in the same manner as the double-size mode shown previously (total the bit values: 4+2+1=7).

The PROGRAM LOOP, lines 3ØØ-37Ø, moves the plane across the screen at the relative speed F (F was set by your input in line 16Ø). Notice in line 32Ø that F is used as the STEP increment-- the plane moves faster by jumping more dots between printing.

To control the slower movement of Sprites 1 and 2, the variable W is decremented from 2Ø to Ø as the program loop is executed. When W=5 the boat moves one dot to the right. When W=Ø the cloud moves one dot to the left.

The "toggling" of the plane is controlled in line 36Ø. By changing the value POKEd in 2Ø4Ø, Sprite Ø is made to alternate between adjacent areas in the sprite table.



If you stare at the landscape long enough while the program runs, peculiar things may begin to happen...flying boats, syntax errors, etc. Like a late-night programmer, this program doesn't know when to quit! Using the counting method of variable W, see if you can insert a successful test to end this program before the boat gets airborn.

SPRITE DEMO

```
10 R=13*4096:REM VIDEO CHIP
20 EN=R+21:POKEEN,0:REM SPRITES DISABLED
30 PRINT"⊐=▓▓▓▓▓▓▓▓▓          CBM 64 SPRITE DEMO"
40 REM***********SET SPRITE VALUES*******
50 X0=R+0:X=1:POKEX0,1:REM PLANE X COORDINATE
60 Y0=R+1:Y=150:POKEY0,Y:REM PLANE Y COORDINATE
70 X1=R+2:J=200:POKEX1,J:REM CLOUD X
80 Y1=R+3:K=90:POKEY1,K:REM CLOUD Y
90 X2=R+4:A=1:POKEX2,A:REM BOAT X
100 Y2=R+5:B=230:POKEY2,B:REM BOAT Y
110 POKER+23,3:REM EXPAND X
120 POKER+29,7:REM EXPAND Y
130 C1=R+39:POKEC1,7:C2=R+40:POKEC2,1:C3=R+41:POKEC3,2:REM SPRITE COLORS
140 POKE 2040,128 :POKE2041,130:POKE2042,131:REM SPR ADDRESS POINTERS,BEGIN 8192
150 FORX=0TO253:READD:POKE8192+X,D:NEXT:REM FILL SPRITE TABLES
160 INPUT"▓▓▓▓        CHOOSE SPEED (1-5)";F
170 REM
180 REM*****PRINT BACKGROUND************
190 PRINT"⊐◥ F1=UP         F3=RANDOM         F5=DOWN"
200 PRINT"▒▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓         ▁▞╱╲╱┐╱╲▞╲▁▔▁"
210 PRINT"◣▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
220 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
230 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
240 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
250 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒◤";:PRINT"▓"
260 REM******SET PROGRAM VALUES********
270 POKE650,128:Z=128:W=4:A=1:POKE2039,253:POKE56310,1:L=R+16
280 POKEEN,7:REM SPRITES ENABLED
290 REM
300 REM********PROGRAM LOOP*************
310 FORM=0TO1:POKEEN,6:POKEL,M:POKEX0,0:POKEEN,7
```

```
320 FORX=1TO255STEPF:POKEX0,X:POKEY0,Y:REM X,Y COORD'S
330 GETC$:Y=Y+(C$=CHR$(133))-(C$=CHR$(135)):IFC$=CHR$(134)THENY=Y-(RND(1)*4-2)
340 W=W-1:IFW=0THENJ=J+1*(J>1)-254*(J=1):POKEX1,J:W=20
350 IFW=5THENA=A+1+255*(A>254):POKEX2,A
360 Z=Z-1*(Z=128)+1*(Z=129):POKE2040,Z
370 NEXT:B=B-1:POKEY2,B:NEXTM:GOTO310
380 REM
390 REM*******SPRITE DATA*************
400 REM
410 REM***SPRITE 0A,PLANE W/PROP******
420 DATA0,255,0,112,36,16,113,255,208,127,255,208
430 DATA127,255,208,0,36,16,0,255,0,0,0,0
440 DATA0,0,0,0,0,0,0,0,0,0,0,0
450 DATA0,0,0,0,0,0,0,0,0,0,0,0
460 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
470 REM***SPRITE 0B, PLANE NO PROP****
480 DATA0,255,0,112,36,0,113,255,192,127,255,208
490 DATA127,255,192,0,36,0,0,255,0,0,0,0
500 DATA0,0,0,0,0,0,0,0,0,0,0,0
510 DATA0,0,0,0,0,0,0,0,0,0,0,0
520 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
530 REM******SPRITE 1, CLOUD***********
540 DATA0,60,0,0,126,0,0,255,16,1,255,184
550 DATA3,255,252,19,255,252,127,255,254,255,255,255
560 DATA255,255,254,127,255,252,15,255,128,7,15,0
570 DATA0,0,0,0,0,0,0,0,0,0,0,0
580 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
590 REM******SPRITE 2, BOAT************
600 DATA0,8,0,0,8,0,0,44,0,0,110,0
610 DATA0,239,0,0,0,0,0,255,128,5,127,0
620 DATA0,0,0,0,0,0,0,0,0,0,0,0
630 DATA0,0,0,0,0,0,0,0,0,0,0,0
640 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

5. M/L Draw Demo

Each of the previous examples has used BASIC line drawing routines, which are slow, as a learning tool. This example contains a moderately fast M/L drawing routine called in lines 210-290. This routine is a "patch-up" from other routines and is not optimized for speed in any way. However, by writing your own BASIC routines to call it, you may be able to realize some of the potential of the '64 Hi-Rez mode. Note that locations 45 and 46 are POKEd in line 100, along with 52 and 56, to protect the M/L program and the BIT-mapped display screen [setup at 8192]. The BIT-mapped screen occupies a full 8K of BYTES: 8192 to 16383.

Note that there is a mode byte set in line 220. Mode 0 = complement (flip) the line; Mode 1 = draw the line; Mode 2 = undraw the line. The complement function leaves the tiny "butterfly" shape behind after each screen, because some portions of the line were "flipped" an even number of times [flip once, it's off; flip again, it's back on].

Finally, note that line 130 sets the screen color, skipping two colors which wouldn't show a trace against the background color. Notice particularly the effect of color values greater than 15.

6. Loading Instructions

1. Turn the '64 OFF to clear and reset machine.

2. Connect the Datasette.

3. Turn power ON, insert cassette and rewind fully.

4. LOAD the desired program by typing (using unshifted keys) the load command and program name (which will appear in uppercase), such as:
   LOAD "3-D DEMO"

5. Press the RETURN key, then press PLAY on the Datasette.

6. The '64 screen will blank for a few moments, then reappear with the name of the first program module, "Character Demo". Now, press the Space Bar, which will either LOAD the module if it is the one you asked for, or will advance the Datasette to the next module. The screen will blank again.
   If the first module is being loaded, the screen will reappear with the cursor and the comment "READY". When "READY" appears, type RUN and press the RETURN key.
   If you are loading the 2nd, 3rd, 4th, or 5th module, each time you press the Space Bar, the Datasette will advance and find the next module, notify you of its name, and proceed when you hit the Space Bar again.
   These steps are repeated as many times as necessary to reach the module you desire and LOAD it. When the screen announces "READY", type RUN and press the RETURN key.

7. If you get a LOAD ERROR, rewind the tape, shut off the '64 momentarily, then repeat Steps 4 through 6.

7. Warranty and Care of Tapes

If you do not get a good load from this or any other tape not made on your system, chances are your unit needs cleaning or alignment. However, if other programs load and you still experience difficulty with our tape, the trouble might be in the tape.

Midwest Micro programs are recorded on high-quality materials using exacting professional standards and the finest high-speed commercial equipment. Yet, strange things can happen to a tape on its way to you.

If you receive a faulty tape, simply return it along with your name and address and a brief description of the trouble. We'll send a new tape promptly, free of charge. Our Warranty is limited to replacement of the program. No other liability is expressed or implied.

If you accidentally damage the tape (for example: erasing it, stepping on it, melt-down, etc.) we will replace it for a small handling charge. Send the original tape, your name and address, and $5.ØØ to:

Midwest Micro Associates
P.O. Box 6148
Kansas City, MO 6411Ø

For your protection, the program is recorded on both sides of the tape. We also recommend that you make a back-up copy of the tape and put it in a safe place. DANGER AREAS include:

magnetic fields (such as stereo equipment, loudspeakers, '64 power transformers, motors, etc.)
heat (such as radiators, heat ducts, window sills, enclosed automobiles, etc.)

NOTES

(Without advancing past leader)

1. CHAR DEMO 0-61 or 62

3. SPRITE DEMO 85-101

5. ML DRAW DEMO 116-152

4. 3D DEMO 101-116 or 117

2. POINT & LINE DEMO 62-84 or 85

6. 152

NOTES

MIDWEST
MICRO associates