Commodore 64 Hi-Res Graphics Made Simple

Paul F. Schatz
www.commodore.ca

One of the Commodore 64's intriguing features is a *high resolution graphics mode*, which divides the screen into 64,000 dots, or *pixels*. By turning these pixels on and off, you can create finely detailed pictures and charts. But because BASIC lacks special graphics commands, only more advanced programmers could use this mode – until now. This article is a breakthrough in that it shows how to add simple graphics commands to BASIC which anyone can use.

Although the high resolution graphics potential of the Commodore 64 is outstanding, accessing and plotting on the hi-res bitmap (320- by 200-pixel resolution) is inefficient and cumbersome from BASIC.

First, BASIC subroutines for calculating and turning on a specific bit can be confusing and intimidating, especially to novice programmers, since the routines require PEEKs, POKEs, ANDs, and ORs. Second, the routines are slow; many BASIC commands need to be interpreted and executed to plot one point. Third, the bitmap has to be located in memory otherwise used by BASIC. The BASIC program space is limited since it is chopped up and some areas are unusable for BASIC programs.

One solution to all of the above shortcomings is to add some new commands to BASIC which drive the high resolution graphics. This article will describe a method for adding four commands.

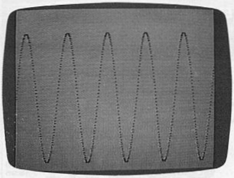
Modifying BASIC

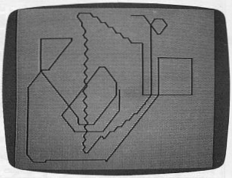
Since there is Random Access Memory (RAM) under the BASIC Read Only Memory (ROM), we can copy an image of BASIC into RAM and then modify it to suit our needs. I have modified BASIC by substituting four new commands, HUE, PLOT, WIPE, and SCREEN, in place of four seldom-used commands, LET, WAIT, CONT, and VERIFY.

Briefly, here's how the new commands were added to BASIC. First, notice that the new keywords are the same length as the keywords they replace. A new keyword has to be mapped exactly into an old keyword's spot in the keyword lookup table. Next, the pointers to the old BASIC routines are changed to point to the routines for the new keywords. Finally, the error message routine is modified so the computer switches to the normal character display if an error is encountered during execution of a program.

A Note To Programmers

The graphing routines were developed with an eye on giving up as little of the BASIC program memory as possible. Not a byte has been lost. This was accomplished by using the RAM memory under the Kernal ROM for the bitmap. Bitmap plotting at this location can only be done properly using machine language routines, since the interrupts have to be turned off and the Kernal ROM switched out to PEEK at the RAM memory. The video matrix, used for the background and foreground color nybbles, is located at $C000 and the machine language graphing routines extend from $C400 to $C545.


A sine wave plotted on the Commodore 64's high resolution graphics screen with Program 2.


Joystick doodles in hi-res graphics with Program 3.

The New Commands

The four new commands, SCREEN, HUE, WIPE, and PLOT, are explained below.

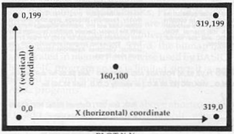
• SCREEN ‹number›
This statement turns on and off the high resolution bitmap. If the number is 1, the bitmap is displayed. If the number is 0, the normal character screen is displayed. Any value other than 1 or 0 will give an ILLEGAL QUANTITY ERROR.

• HUE ‹number›, ‹number›
This statement determines the colors displayed on the bitmap. The first number defines the foreground color (color displayed for bits set to 1). The second number defines the background color. A number 16 or greater will give an ILLEGAL QUANTITY ERROR. The color codes are:

0 Black	4 Purple	8 Orange	12 Gray2
1 White	5 Green	9 Brown	13 Light Green
2 Red	6 Blue	10 Light Red	14 Light Blue
3 Cyan	7 Yellow	11 Gray1	15 Gray3

• WIPE
This statement causes a high-speed clear of the bitmap. All the bits are set to zero and the screen is cleared.

Coordinates For PLOT



PLOT X,Y

• PLOT ‹number›, ‹number›
This statement sets a bit on the bitmap, causing the corresponding pixel on the screen to be displayed in the foreground color. A coordinate system with an origin (0, 0) at the lower-left corner is used (see the figure). The first number is the horizontal position relative to the origin, and the second number is the vertical position relative to the origin. The first number can have values from 0 to 319, and the second number can have numbers from 0 to 199. Numbers outside these ranges give an ILLEGAL QUANTITY ERROR.

Loading In The New BASIC

The new BASIC is loaded into the Commodore 64 by entering and running Program 1. When entering the program, be accurate, since an incorrect number may cause the computer to crash (forcing you to switch it off and on to clear it). To be safe, SAVE the program before running it for the first time. A checksum is included to warn if there is a mistake somewhere in the DATA statements. It will take the computer a minute or two to run the program. To enable the new BASIC, enter:

```
POKE 1,54
```

The new BASIC can be disabled by pressing the RUN/STOP and RESTORE keys simultaneously or by entering:

```
POKE 1,55
```

When entering programs using the new graphics commands, the new BASIC must be enabled so the tokenizing routine will recognize them. The commands they replaced will no longer work unless the new BASIC is disabled.

Some Simple Programs

We are now ready to enter and run a couple of simple programs using the new BASIC. First, a simple sine wave. Load and [illegible] the new BASIC, type NEW, switch on the new BASIC, and enter Program 2.

Now type RUN and watch the sine wave appear. Wasn't that easy? Compare this program with the one in the *Commodore 64 Programmer's Reference Guide* (pp. 122-26) for ease of programming and speed of execution.

Now, how about a joystick-driven doodle pad? Be sure Program 2 is saved. Then type NEW and enter Program 3. Plug a joystick into port two.

Only The Beginning

Programs written with the new BASIC can be loaded and saved in the normal fashion (but remember, we did away with VERIFY). My purpose was to provide a useful rudimentary graphing tool and to demonstrate the ease with which BASIC can be modified to include new commands. There are numerous extensions of both aspects which could be implemented. For example, a high-speed line drawing command, LINE; or a new command similar to the ON-GOTO statement but with the branching determined by the joystick position, i.e., JOYGOTO, or JOYGOSUB....

See program listings on page 111.

Hi-Res Graphics (Commodore 64)

BEFORE TYPING...
Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs" and "A Beginner's Guide To Typing In Programs" that appear before the Program Listings.

Program 1: New BASIC

```
0 REM BASIC HI-RES
10 A=0:REM INTIALIZE CHECKSUM
20 REM MOVE BASIC ROM TO RAM
30 FORI=40960TO49151:POKEI,PEEK(I):NEXTI
40 REM CHANGE LET TO HUE
50 FORI=41150TO41152:READN:POKEI,N:A=A+N
   :NEXTI
60 READL,H:POKE40988,L:POKE40989,H:A=A+L
   +H
70 DATA 72, 85, 197, 75, 196
80 REM CHANGE WAIT TO PLOT
90 FOR I=41189TO41192:READN:POKEI,N:A=A+
   N:NEXTI
100 READL,H:POKE41008,L:POKE41009,H:A=A+
    L+H
110 DATA 80, 76, 79, 212, 130, 196
120 REM CHANGE CONT TO WIPE
130 FORI=41225TO41228:READN:POKEI,N:A=A+
    N:NEXTI
140 READL,H:POKE41024,L:POKE41025,H:A=A+
    L+H
150 DATA 87, 73, 80, 197, 53, 196
160 REM CHANGE VERIFY TO SCREEN
170 FORI=41201TO41206:READN:POKEI,N:A=A+
    N:NEXTI
180 READL,H:POKE41014,L:POKE41015,H:A=A+
    L+H
190 DATA 83,67,82,69,69,206,11,196
200 REM CHANGE ERROR MESSAGE ROUTINE
210 FORI=42042TO42044:READN:POKEI,N:A=A+
    N:NEXTI
220 DATA 76, 0, 196
230 REM READ IN NEW ROUTINES
240 FORI=50176TO50480:READN:POKEI,N:A=A+
    N:NEXTI
250 IFA<>39040THENPRINT"ERROR IN DATA ST
    ATEMENTS"
260 END
300 DATA 32, 24,196,138, 10,170, 76, 61,
    164, 80, 70, 83, 32,158,183,224, 1
310 DATA144, 5,240, 19, 76, 72,178,169,
    27,141, 17,208,169, 21,141, 24,208
320 DATA169,151,141, 0,221, 96,169, 59,1
    41, 17,208,169, 8,141, 24,208,169
330 DATA148,208,238,162, 32,169,224,133,
    252,160, 0,132,251,152,145,251,200
340 DATA208,251,230,252,202,208,246, 96,
     32,123,196,138, 10, 10, 10, 10,133
350 DATA 2, 32,253,174, 32,123,196,138,
    5, 2,160,192,132,252,160, 0,132
360 DATA251,162, 2,145,251,200,208,251,2
    30,252,202, 16,246,145,251,200,192
370 DATA232,144,249, 96, 32,158,183,224,
     16,176, 17, 96, 32,235,183,134, 2
380 DATA169,199, 56,229, 2,133, 2,201,20
    0,144, 3, 76, 72,178,165, 21,240
390 DATA 10,201, 1,208,245,165, 20,201,
    64,176,239,169, 0,133,251,169,224
400 DATA133,252,165, 20, 41,248, 24,101,
    251,133,251,165, 21,101,252,133,252
410 DATA165, 2, 41, 7, 24,101,251,133,25
    1,144, 2,230,252,165, 2, 74, 74
420 DATA 74, 10,170,189,247,196, 24,101,
    251,133,251,189,248,196,101,252,133
430 DATA252,165, 20, 41, 7,170,160, 0,12
    0,169, 52,133, 1,177,251, 29, 41
440 DATA197,145,251,169, 54,133, 1, 88,
    96, 0, 0, 64, 1,128, 2,192, 3
450 DATA 0, 5, 64, 6,128, 7,192, 8, 0, 1
    0, 64, 11,128, 12,192, 13, 0
460 DATA 15, 64, 16,128, 17,192, 18, 0,
    20, 64, 21,128, 22,192, 23, 0, 25
470 DATA 64, 26,128, 27,192, 28, 0, 30,1
    28, 64, 32, 16, 8, 4, 2, 1
```

Program 2: A Simple Sine Wave

```
10 SCREEN 1: REM TURN ON BITMAP
20 WIPE: REM CLEAR BITMAP
30 HUE 0,1: REM BLACK DOTS, WHITE SCREEN
40 FOR X=0 TO 319 STEP .5
50 Y=INT(90+80*SIN(X/10))
60 PLOT X,Y: REM PLOT POINT
70 NEXT X
80 GET A$: IF A$="" THEN 80: REM WAIT FO
   R KEYSTROKE
90 SCREEN 0: REM NORMAL SCREEN
```

Program 3: A Joystick-Driven Doodle Pad

```
10 SCREEN 1: WIPE: HUE 0,1
20 X=159: Y=99: PLOT X,Y
30 GOSUB 100: IF J=15 THEN 30
40 PLOT X,Y: GOTO 30
50 SCREEN 0: END: REM GRACEFUL EXIT
100 REM READ JOYSTICK
110 J=PEEK(56320) AND 15: REM PORT 2
120 IF (J AND 8)=0 THEN X=X+1: REM MOVE
    RIGHT
130 IF (J AND 4)=0 THEN X=X-1: REM MOVE
    LEFT
140 IF (J AND 2)=0 THEN Y=Y-1: REM MOVE
    DOWN
150 IF (J AND 1)=0 THEN Y=Y+1: REM MOVE
    UP
160 IF Y<0 THEN Y=0: REM STAY IN RANGE
170 IF Y>199 THEN Y=199
180 IF X>319 THEN X=319
190 IF X<0 THEN X=0
200 GET A$:IF A$=CHR$(147) THEN WIPE: RE
    M CLEAR SCREEN
210 IF A$=CHR$(136) THEN 50: REM F7 KEY
    TO EXIT
220 RETURN
```