GFXterm - VT100/ANSI Terminal Emulation with Graphics Support

Overview:

GFXterm is a simple terminal emulator designed for use with Geoff Graham's single-chip Micromite computers running MMbasic. As such, it provides just enough VT100/ANSI emulation to use the Micromite's inbuilt editor with the default 80 column by 24 line screen size.

In addition, GFXterm supports a set of graphics extensions that are suitable for drawing simple charts, diagrams, and rolling graphs; lines and arcs can be drawn, enclosed regions filled, and rectangular areas scrolled in any direction. Graphics are drawn on a separate 'glass layer' overlaying the normal text screen. This layer is turned off be default, only being turned on once a graphics command is received. The graphics layer can then be turned off again by pressing **alt-C** to clear all graphics. GFXterm runs slightly faster with graphics turned off.

Text and graphics layers operate completely independently of each other, and do not in any way interact, with the text layer visible through 'clear' areas of the graphics layer (wherever pixels are set to the colour: R=0, G=0, B=0).

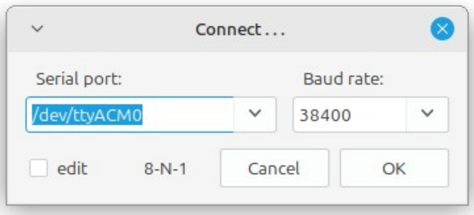
The latest versions of GFXterm (2021 onwards) have been ported to Lazarus/FPC, and successfully compiled for WINDOWS, LINUX, and RASPBIAN. With minimal changes it should also be possible to compile for MACOS (the operating system formerly known as OS X).

Operation:

The only thing needed to run GFXterm is a single executable file, generally named GFXtermW32.exe (WINDOWS), or GFXterm32/64 (32 or 64-bit LINUX and RASPBIAN). Just double-click on the icon to run this file.

Upon start-up you will be presented with an empty terminal window and flashing red block cursor at the top left. At this point the terminal is sitting in the disconnected (local loopback test) state. Any ASCII keypresses will be printed to the screen and control characters processed, while function and cursor keys will display their escape sequences as plain-text. For example, pressing the **F1** key will display `esc"[11~"`. Pressing **ENTER** will move the cursor to the first column, pressing **ctrl-ENTER** will move the cursor vertically down one line. Pressing **alt-ENTER** will do both - moving the cursor to the beginning of the next line.

To connect to an attached Micromite, right-click anywhere on the terminal window (or press **alt-M**), and from the pop-up menu that appears select CONNECT. A dialog window will then appear, prompting you to select a serial port and baud rate (note that by default the Micromite console runs at 38400 baud). Once you make the required selection, press **ENTER** or click the OK button, and you should be connected to your Micromite.



The pop-up menu is the main method of controlling GFXterm, although a number of the functions available in the menu are also mapped to shortcut (**alt-**) keys. There are also a few functions that are only accessible through shortcut keys.

Menu Commands:

CONNECT / DISCONNECT - used to connect to or disconnect from a Micromite. The serial port and baud rate are selected from two drop-down menus. Other serial port parameters default to 8 data bits, no parity, 1 stop bit, which if needsbe can be changed by right-clicking on the 8-N-1 label and selecting alternatives as required. An edit check-box is provided to allow manually entering the port name/path in case the required serial port is not offered as a selection (in theory, this feature should never be needed).

NOTE: Upon connecting, GFXterm will assert the DTR modem control line. This is to keep an attached Raspberry Pi Pico (RP2040/2350) that is running MMbasic happy, but may also cause an Arduino device to reset – the Arduino design makes use of asserting DTR to initiate firmware upload.

Normally GFXterm remembers the last successful serial connection; this connection can be restored by simply pressing **ctrl-SPACE**, without needing to go through the menu.

LOG to file / STOP logging - used to save terminal output from the Micromite to a text file. Only plain text is saved, without any colour or other formatting. Normally, logging will be used to record program output, or to save a program held within the Micromite to your PC by typing **LIST ALL** after having started logging. The STOP logging function is also mapped to the **alt-L** key combination, with a second press of **alt-L** then resuming logging and appending to the end of the previous log file. To log to a different file, instead of using **alt-L** select Log to file from the pop-up menu and enter a new log file name.

Select / Copy Text - this displays a monochrome text-only view of the terminal screen from which it is then possible to select text with the mouse and copy it to your computer's clipboard using **ctrl-C**. The view is a frozen snapshot, with the remainder of GFXterm continuing to operate in the background. Once finished copying, press **ENTER** or **ESC** to exit this view.

Paste (from Clipboard / from Text File) - these two options can be used to upload a program to the Micromite. GFXterm detects if pasting into the Micromite's inbuilt editor and slows down the speed of pasting to accommodate. While at the MMbasic command prompt, you can first type **AUTOSAVE** to quickly save a program directly to the Micromite's flash memory. Pasting from the clipboard is also mapped to the **alt-P** key combination.

HINT: if pasting into the Micromite's inbuilt editor, always ensure there is a space character to the right of the cursor before you start in order to suppress automatic line indenting.

CANCEL Paste - immediately cancels any paste operation that is in progress, in case of inadvertently pasting from an unintended source. This function is also mapped to the **alt-Z** key combination.

Screen Font - select from available monospaced fonts and sizes available on the computer. The default font and size (for Linux) is “Monospace” 12 point. This dialog also allows you to tweak a number of other font settings, but in most cases this is not necessary. One setting that may be of interest, however, is the / zero check-box, that for some fonts will change the display of the digit zero to have a forward-slash through it. When closed, GFXterm remembers new font settings.

NOTE: changing the font family and/or size will also alter the horizontal and vertical pixel counts for graphics - so these values should never be assumed. Instead, the horizontal and vertical counts should always be read back from GFXterm immediately before using any graphics commands (see: GFX “?” command).

Screen Size – the minimum size is 40x16, while the maximum is 160x60. The default is 80x24. Ideally the number of columns should be kept a multiple of 8, to allow for TABs to be handled nicely when lines overflow. When closed, GFXterm remembers the new screen size setting.

Font Colour (Red, Green, Yellow, Blue, Magenta, Cyan, WHITE, swap B/W) - selects the default text colour. The default colour setting is WHITE, but the Micromite can always override this setting, as happens when the inbuilt editor is used with **OPTION COLOURCODE ON**. swap B/W is black text on a white background.

Dimmable Text (enabled, bright #1, bright #2) - determines how the **SGR 2** (dim foreground) escape sequence is handled. When enabled (default), the foreground colour can be selected between bright and dim with **SGR 2** while the background colour is always dim. Selecting bright #1 overrides **SGR 2**, forcing foreground colours to always be bright, while non-black background colours are also forced to bright when bright #2 is selected.

Palette Editor – this dialog allows any of the 16 available text palette colours to be edited to suit personal preferences. For example, the entry for Black can be changed to be a light grey (R=32, G=32, B=32) if you find a pure-black terminal background uncomfortable to view. Changes are saved upon exiting GFXterm.

Clear / Reset (GFX layer, Text layer, Ring buffer, RESET terminal) - these commands are also mapped to the keys **alt-C**, **alt-D**, **alt-A** and **alt-R** respectively. The most useful one of these will be **alt-C** for where a program has drawn on the graphics layer, and you want to clear this so as to see underlying text, such as when going back into the Micromite's inbuilt editor after a program that uses graphics has ended.

Diagnostics - output is displayed within a separate text window as supported by the operating system. To use these options under LINUX, it is necessary to start GFXterm from the command line of a Linux Terminal. Under WINDOWS, GFXterm can just be started up normally, and will automatically open up a Command Console itself when one of the below diagnostics options is selected.

- Rx Data -> console

For each packet of data received, a timestamp and packet size is displayed on the console, followed on subsequent lines by a plain-text version of the data with decoded hex values for control characters and the top 128 non-ASCII character codes. Control and top 128 codes are displayed with a highlighted background, using an alternating pair of colours to improve readability. For example:

```
03:36:52.709ms   55 bytes
<ESC>[37m <ESC>[36mElse<ESC>[37m phase<ESC>[37m=<ESC>[32m
6<ESC>[K<ESC>[37m<CR><LF><ESC>[16;1H
```

- VT and GFX codes

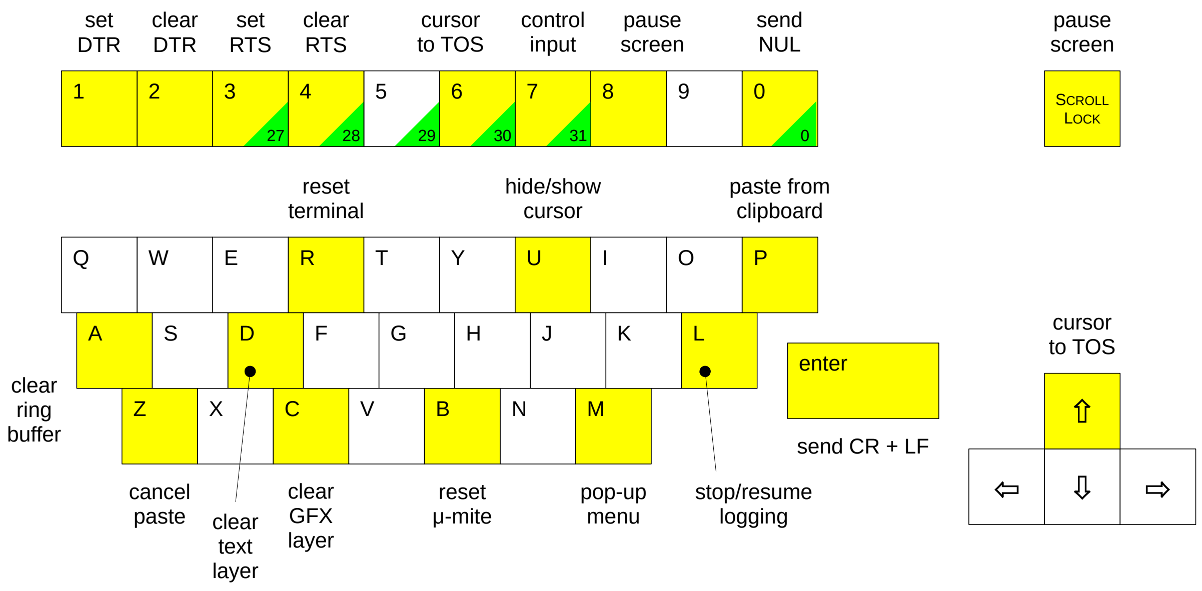
Displays a running list of VT and GFX commands processed, along with their pass/fail status. Codes that can not be decoded by GFXterm are highlighted with a red background. This information can be extremely useful for identifying invalid or unsupported VT code sequences being generated by an attached device.

About and Exit – the About dialog provides build and version information, plus includes a button to generate the PDF of this help file on the Desktop. Exit closes GFXterm.

Keyboard Shortcuts:

alt-Z	CANCEL any paste operation that is in progress	*(also a menu item)*
alt-A	clear ring buffer (256k buffer used for serial input)	*(also a menu item)*
alt-C	clear and turn off graphics layer	*(also a menu item)*
alt-D	clear text layer	*(also a menu item)*
alt-P	paste from clipboard	*(also a menu item)*
alt-L	STOP/RESUME logging	*(also a menu item)*
alt-R	RESET terminal	*(also a menu item)*
alt-> AND **<**	bell volume level up and down	
alt-U	hide/show cursor	
alt-B	send break, signals 1455 firmware to reset Micromite	
alt-M	show pop-up menu (in case you have no mouse)	
alt-ENTER	send CR + LF character sequence	
alt-ZERO	send null character (also **ctrl-ZERO** for VT102 compatibility)	
alt-1 TO **4** *	set/clear modem control (DTR/RTS) outputs	
alt-6 *	scroll screen and cursor up, until the cursor is at the top line *(also mapped to the* **alt-⇧** *key combination)*	
alt-7 *	control input, enter mixed text and control codes to send as a string	
alt-8 *	pause output to screen, serial input is buffered until un-paused *(also mapped to the* **ScrLock** *key)*	
ctrl-SPACE	try to restore last valid connection, or disconnect if already connected	
ctrl-3 TO **7**	generate the control characters between 27_{10} and 31_{10} *(alternatives to* ***ctrl-[, ctrl-\, ctrl-], ctrl-~*** *OR* ***ctrl-^, ctrl-/*** *OR* ***ctrl-_*** *)*	

* some operating systems hook **alt-n** (0..9) key combinations, to work around this **ctrl-alt-n** is also mapped to the same functions and can be used instead



Status Line:

Above the terminal text area is a status line that displays various useful pieces of information about the terminal session. This includes the hex codes for the last ASCII character displayed(1) and key pressed(4), current cursor location(2,3), last Rx and Tx time information(5,6), and various status indicators(7-11) :

```
00  row=01  col=001  key=00  00.0s  00.0s  00%  online  logging  [000000]  DTR RTS CTS dsr
1.     2.       3.       4.     5.      6.    7.     8.       9.        10.           11.
```

1. last ASCII character displayed (Rx)
2. cursor position, current row
3. cursor position, current column
4. last ASCII key pressed (Tx)

5. time since last serial data was received
6. time since last serial data was transmitted

7. percentage of Rx ring buffer in use:

silver	<10% used
blue	10% to 39% used
yellow	40% to 69% used
red	>70% used

8. online status annunciator (alt-8 or scroll-lock toggles pause):

green	connected to serial port
green / yellow (flashing)	connected, but screen updates paused

9. logging annunciator - yellow when logging Rx data to a file

10. number of characters waiting to be pasted:

green	fast paste
blue	slow paste

11. status of modem control lines. DTR and RTS are outputs that are controlled from the keyboard, while CTS and DSR are inputs from an attached serial device:

UPPERCASE	signal line is set / asserted
lowercase	signal line is clear

XModem Transfers (MMbasic specific):

BASIC programs can be loaded from a local file into the Micromite's memory by pressing the **F11** function key when sitting idle at the MMbasic command prompt. A **Load File** dialog window will pop up from which the file to load from should be selected. After this GFXterm will manage the upload process. When loading has been completed, the command prompt will be returned to.

Likewise, BASIC programs can be saved from the Micromite's memory to a local file by pressing the **F12** function key when sitting idle at the MMbasic command prompt. GFXterm will manage the download process, after which a **Save File** dialog will pop up to select the local file to save the data to. After saving, the command prompt will be returned to.

A running counter is displayed within the terminal window while any XModem transfer is in progress. This transfer can be cancelled by pressing **ctrl-C** at any time. Doing so before a file has finished loading will mean any program previously held in the Micromite's memory should be left intact – the Micromite buffers to RAM first, only writing a copy to flash once the whole program has loaded.

VT and ANSI Commands:

See Appendix A for a full list of the commands that have been implemented (marked with ticks). The list mostly covers the basic VT100 and VT102 commands, with a few VT220 specific additions and ANSI colour support. Note that VT52 compatibility mode has not been implemented.

Colour for text and background (using **SGR**) is supported, with 8 possible colours for each. Background colours are dimmed, while foreground colours are bright by default. However, a dim foreground attribute (**SGR 2**) is available to effectively give 16 foreground colours if dimmable text has not been disabled. The blinking attribute (**SGR 5**) is ignored.

A vertical scroll region can be set with **DECSTBM**, but only **IND**, **NEL**, **RI**, **CUU**, **CUD**, **IL**, **DL**, and **LF** make use of the top/bottom margins set. Character delete/insert/erase is supported with **DCH**, **ICH** and **ECH,** while character insert/replace mode is selectable via **RM** / **SM 4**. Setting and clearing tab stops is not supported, with tab stops instead fixed at every 8 columns.

AutoWrap at end-of-line is hard-wired on, and cursor positioning is permanently fixed to absolute - location (1,1) being top-left of the terminal screen irrespective of any top or bottom margins set with **DECSTBM**.

Mouse scroll wheel activity is mapped to the cursor up/down keys (unshifted) and cursor left/right keys (shifted), and will work with the Micromite's internal editor. Further to this, mouse position reporting (X10, VT200, and extensions thereof) can be turned on/off with **SM** / **RM ?9**, **?1000**, **?1006**, and **?1015**, thereby providing simple mouse support via single-click (X10) and button down/up (VT200) pointer location only - no drag-and-drop, mouse tracking, etc. Modifiers of SHIFT, CTRL, and ALT are detectable in the VT200 mouse mode.

The text cursor can be hidden/shown using **RM** / **SM ?25** - this may also be manually controlled from the keyboard using **alt-U** to toggle the cursor state.

GFX Commands:

The syntax of a GFX command string is as follows:

<**DLE**> **`command parameter,..., parameter`** <**CR**> [<**LF**>]

where the command and parameters (all in plain text) can be separated by spaces, commas, semicolons, or tab characters. A GFX command string is terminated by a carriage return, with any immediately following line feed skipped. **<DLE>** (data link escape) is **`chr$(16)`**.

For example, to draw a circle with centre at (100,100), radius 50, coloured green using a brush 3 pixels wide, the following lines of MMbasic code would be used:

```
PRINT chr$(16) "i" 0, 255, 0, 3
PRINT chr$(16) "a" 50, 50, 150, 150, 0, 0
```

Commands are detailed in Appendix B. They can be written in full, or abbreviated to the first letter, and can be upper or lower case. The best way to understand how to use GFX commands is to look at the two sample programs in Appendix C: "`GFX demo.bas`" and "`GFX bouncing ball.bas`". The commands are, on the whole, just wrappers for graphic object commands provided by the operating system.

The following points are worth noting:

1. The horizontal and vertical pixel counts are dependant on the font size and number of rows and columns selected in GFXterm, hence any program using graphics should first issue the "**?**" command to retrieve these counts and scale output accordingly.

2. The origin for all graphics commands is the bottom left corner of the terminal window.

3. The **`ink`** command accepts red, green, and blue parameters with each ranging from 0 to 255. An ink colour of 0,0,0 is transparent - for opaque black use 1,1,1 instead.

4. The **`arc`** command can draw circles, ellipses, or parts thereof. The angles specified are in degrees, with 0 degrees due north (12 o'clock), positive values moving clockwise - set start and finish as both 0 for a full circle or ellipse.

5. The **`fill`** command expects a fully enclosed area bounded by the current ink colour, flood filling the enclosed region with that same colour. You can not fill with a different colour. Unfortunately, while generating no error, **`fill`** does not function on LINUX systems.

To prevent overflowing the serial input ring buffer, a program can synchronize with GFXterm using the control codes **<ENQ>** and **<ACK>** (**`chr$(5)`** and **`chr$(6)`** respectively). When GFXterm sees an **<ENQ>** as it processes incoming serial data, it responds by sending an **<ACK>** in reply. This allows a running MMbasic program to wait for GFXterm to catch up.

Other Notes:

GFXterm has an internal 256k ring buffer that holds incoming serial data before it is displayed on-screen. In normal use only a very small portion of this ring buffer is ever used, however under certain extreme circumstances it is possible to fill the buffer. For instance, an MMbasic program that sits in a tight loop rapidly outputting data for an extended period of time, combined with manually selecting a non-bitmapped screen font (non-bitmapped fonts take considerably more time to display on the screen).

When the ring buffer becomes more than 98% full, GFXterm responds by suspending the printing of characters onto to the screen - the cursor position still updates, and the screen still scrolls as normal, but no characters appear. When the ring buffer drops below 95% full, character printing resumes. In most cases this should prevent the ring buffer ever reaching 100% full (at which point incoming serial data is simply discarded).

Remember: the keyboard shortcut **alt-C** *clears the graphics layer - when creating scrolling graphs this can be extremely useful to allow viewing of the text underneath.*

Robert Rozée
7-February-2025

Appendix A:

VT100/102 Command Set

	Character:	Action:
☑	BELL (7)	Beeps the terminal
☑	BS (8)	Moves the cursor back one column
☑	HT (9)	Moves the cursor to the next tab stop
		(tabstops are fixed at every 8th column)
☑	LF (10)	Moves the cursor down one row
	VT (11)	*(same as LF)*
	FF (12)	*(same as LF)*
☑	CR (13)	Moves the cursor to column one
☒	CAN (24)	Cancels an Escape sequence.
☑	ESC (27)	Starts an Escape Sequence

Escape Sequences:

	NAME	DESCRIPTION	CODE SEQUENCE
☑	CUU	Cursor Up	ESC [Pn A
☑	CUD	Cursor Down	ESC [Pn B
☑	CUF	Cursor Forward	ESC [Pn C
☑	CUB	Cursor Backward	ESC [Pn D
☑	CUP	Cursor Position	ESC [Pn ; Pn H
☑	HVP	Horizontal and Vertical Position	ESC [Pn ; Pn f
☑	IND	Index	ESC D
☑	NEL	Next Line	ESC E
☒	HTS	Horizontal Tabulation Set	ESC H
☑	RI	Reverse Index	ESC M
☑	DECSTBM	Set Top and Bottom Margin	ESC [Pn ; Pn r
		(used by IND, NEL, RI, CUU, CUD, DL, IL, and LF/VT/FF)	
☑	DECSAVC	Save Cursor and attributes	ESC 7
☑	DECRESC	Restore Cursor and attributes	ESC 8
☑	ED	Erase in Display *(ignores scroll region)*	ESC [Ps J
		0 - cursor to EOD	
		1 - BOD to cursor	
		2 - full display ~~(ansi: cursor to 1,1)~~	
☑	EL	Erase in Line	ESC [Ps K
		0 - cursor to EOL	
		1 - BOL to cursor	
		2 - full line	
☑	DL	Delete Line	ESC [Pn M
☑	IL	Insert Line	ESC [Pn L
☑	DCH	Delete Character	ESC [Pn P
☑	ICH	Insert Character (VT200)	ESC [Pn @
☑	ECH	Erase Character (VT200)	ESC [Pn X
☒	TBC	Tab Clear	ESC [Ps g
		0 - clear tab at current position	
		3 - clear all tab stops	

☒	DECKPAM	Keypad Numeric Mode	ESC >
☒	DECKPNM	Keypad Application Mode	ESC =
☑	RM	Reset Mode	ESC [Ps l
		(same Ps values as SM, below)	
☑	SM	Set Mode	ESC [Ps h
☑	IRM	4 Insert/Replacement mode	
☒	LNM	20 Line feed/New line Mode	
☒	DECCKM	? 1 Cursor/application Keypad Mode	
☒	DECOM	? 6 Origin/absolute Mode	
☒	DECAWM	? 7 AutoWrap at end-of-line Mode	
☑		? 9 Enable X10 Mouse reporting	
☑		? 1000 Enable VT200 Mouse reporting	
☑		? 1006 Use SGR encoding	
☑		? 1015 Use URXVT encoding	
☑		? 25 Show Text Cursor (use RM to hide)	
☒	SCS	Select Character Set	ESC (Px
		A - ASCII Set	
		B - ASCII Set	
		0 - Special Graphics	
☑	SGR	Select Graphic Rendition	ESC [Ps m
		0 - normal	
		1 - bold	
		2 - dim foreground	
		4 - underlined	
☒		5 - blinking (not supported)	
		7 - reverse video	
		30 to 37 - foreground colour	
		39 - use default foreground colour	
		40 to 47 - background colour	
		49 - use default background colour	
☒	RIS	Reset to Initial State	ESC c
☒	DECSTR	Soft Reset	ESC [! p
☑	DSR	Get Cursor Position	ESC [6 n
		terminal responds with:	
		"ESC [Pn ; Pn R"	

Appendix B:

GFX Commands

command:	abbreviation:	param 1:	param 2:	param 3:	param 4:	param 5:	param 6:
?	**?**	<width>	<height>	←(these parameters returned to target)			
clear	**C**	x_1	y_1	x_2	y_2		
ink	**I**	R	G	B	width		
line	**L**	x_1	y_1	x_2	y_2		
plot	**P**	x	y				
arc	**A**	x_1	y_1	x_2	y_2	$angle_1$	$angle_2$
fill	**F**	x	y				
moveto	**M**	x	y				
drawto	**D**	x	y				
scroll	**S**	x_1	y_1	x_2	y_2	$delta_X$	$delta_Y$

GFX commands draw using the RGB colour set with **ink**, using the specified (param 4) line width. A virtual 'pen' is used for all drawing commands (**line**, **plot**, **arc**, **drawto**) , with the pen's location remaining at the end point of the last drawing command. The **moveto** command just moves the pen position, without placing anything on screen.

In cases where two pairs of x,y parameters are specified, (**x_1**,**y_1**) defines one corner of an area, while (**x_2**,**y_2**) defines the diagonally opposite corner. The commands **clear**, **arc**, and **scroll** operate within this area, while the **line** command draws a straight line between the two specified points. In the case of **arc**, the rotational centre is located in the centre of the specified area - if the area is non-square then an oval (or part thereof) will be drawn.

The command **plot** sets the colour of a dot centred at (**x**,**y**) and of size specified by the **ink** command, while **moveto** just moves the pen position without drawing anything and **drawto** draws a straight line from the last pen position to the specified (**x**,**y**) coordinate.

Unfortunately, while generating no error, **fill** does not function on Linux systems; on other systems it performs a flood fill of an area containing the point (**x**,**y**) and fully enclosed by pixels matching the current ink colour, using that same colour for the fill.

Appendix C:

GFX Demo.bas

```
Const GFX=Chr$(16)
Const ENQ=Chr$(5)
Const ACK=Chr$(6)

Print GFX "?"
Input Gw, Gh

Print GFX "clear" 0, 0, Gw, Gh
Print GFX "ink" 255, 255, 0, 7

angle=0
Y0=Int(Gh/2)
T0%=Timer

phase=1
R=&hFF
G=0
B=0

Do
  angle=(angle+2) Mod 360
  value=Sin(angle*Pi/180)
  Y1=Int(Gh/2-value*Gh/3)

  If phase=1 Then
    If G<255 Then G=G+3 Else phase=2
  ElseIf phase=2 Then
    If R>0   Then R=R-3 Else phase=3
  ElseIf phase=3 Then
    If B<255 Then B=B+3 Else phase=4
  ElseIf phase=4 Then
    If G>0   Then G=G-3 Else phase=5
  ElseIf phase=5 Then
    If R<255 Then R=R+3 Else phase=6
  Else
    If B>0   Then B=B-3 Else phase=1
  EndIf

  R=Min(Max(0, R), 255)
  G=Min(Max(0, G), 255)
  B=Min(Max(0, B), 255)

  Print GFX "ink" R, G , B, 7
  CV%=(((Int(R) << 8) Or Int(G)) << 8) Or Int(B)

  Print GFX "scroll" 0, 0, Gw-1, Gh-1, -1, 0
  Print GFX "line" Gw-10, Y0, Gw-9, Y1
  Y0=Y1

  T1%=Timer
  Print , angle, Y1, Str$(value, -3, 3),, T1%-T0% "ms",, Hex$(CV%, 6)
  T0%=T1%

  Do :Loop Until (Inkey$=ACK) Or (Timer-T1%>500)
  Print ENQ;
Loop
```

Appendix C (continued)

GFX Bouncing Ball.bas

```
Const GFX=Chr$(16)

Print GFX "?"
Input Gw, Gh

Print GFX "clear" 0, 0, Gw, Gh

X1=Gw\2-50
Y1=Gh\2-50
X2=Gw\2+50
Y2=Gh\2+50

Print GFX "ink" 255, 255, 0, 5
Print GFX "arc" X1+5, Y1+5, X2-5, Y2-5, 0, 0

Print GFX "ink" 255, 0, 0, 2
Print GFX "arc" Gw\2-30, Gh\2-20, Gw\2-10, Gh\2-10, 0, 0
Print GFX "arc" Gw\2+10, Gh\2-20, Gw\2+30, Gh\2-10, 0, 0
Print GFX "fill" Gw\2-20, Gh\2-15
Print GFX "fill" Gw\2+20, Gh\2-15

Print GFX "ink" 0, 255, 0, 5
Print GFX "moveto" Gw\2, Gh\2-5
Print GFX "drawto" Gw\2+5, Gh\2+8
Print GFX "drawto" Gw\2, Gh\2+8

Print GFX "ink" 0, 0, 255, 5
Print GFX "arc" Gw\2-25, Gh\2, Gw\2+25, Gh\2+30, 100, 260

dX=-1
dY=-1

Do
  Print GFX "S" X1, Y1, X2, Y2, dX, dY
  Print, X1, Y1, X2, Y2, dX, dY
  X1=X1+dX
  Y1=Y1+dY
  X2=X2+dX
  Y2=Y2+dY

  If X1=0 Then dX=-dX
  If Y1=0 Then dY=-dY
  If X2=Gw-1 Then dX=-dX
  If Y2=Gh-1 Then dY=-dY

  Pause 10
Loop
```