Pixel Graphics for Blue Label Pascal

Introduction

The Nascom computer comes as standard with a character set which includes a set of ‘graphics blocks’. Each character is three blocks high by two blocks wide. This means that a ‘graphics’ screen measures 240 blocks high by 96 blocks wide. This is OK for space invaders and other simple games but a bit limiting for maths and other applications requiring a little more detail.

Each display character measures 8 pixels wide by 16 pixels high. If the top line of the display is included, that gives a graphic screen measuring 256 pixels high by 384 pixels wide. A worthwhile gain for some applications - back in 1983, it seemed pretty good.

This pixel plot feature depends on:-

1. The display being fitted with programmable character generator hardware
2. The VGRA.OV overlay being present on the Polydos master (boot) drive

The programmable character set is limited to 256 characters maximum. If all are given over to ‘pixel drawing’, they would cover approximately 1/3 of the screen (16 x 48 characters is 768 characters) It’s quite usual to set aside less than 256 characters for plotting, so that a mixture of capital letters and plotting may be displayed.

Basic Principle

When a plot command is executed, the screen coordinates are converted to a character position. If this position is on-screen, then a programmable character is taken from a reserve list, set to the background pattern and placed at that location. The required pixel is then switched on. If a programmable character is already at that location, then another pixel is turned on in that character. This is repeated for further points, lines or circles.

The number of programmable characters left in the list can be read. If the number drops to zero, then all the programmable chars have been used somewhere on the screen. They will continue to get written too, but no new character positions will be created.

A better idea is to limit plotting to complete entirely before the list is used up. Before the next cycle of plotting begins, a FIXFRAME command can be issued to signal that all programmable graphics are available to be reused.

An animated display can be produced by FIXFRAME, plot the display, FIXFRAME, change the plots, plot the display, FIXFRAME ... and so on.

Graphics address

The programmable graphics in a Nascom 1-3 are assumed to be located in shadow RAM located behind main memory starting at E000H. That is, writing to E000 to E800 changes the character set. Each character is defined by 16 bytes.

If a Nascom 4 is detected during initialisation, then that slightly different programmable graphics system is adopted automatically.

If your graphics chars are located at another start address, edit the value of GRAPHICZERO to suit in VGRA.TX, then re-assemble the overlay using the command:-

PZAP VGRA.TX VGRA.OV

VGRA.OV needs to be stored on the master disk so the system can use it.

Plotting Range

Y axis - 0 → 255 is visible on screen, in the range -32768 → 32767

X axis - 0 → 383 is visible on screen, in the range -32768 → 32767

The point 0,0 is the bottom left corner of the screen.

All 16 lines of the display are available for plotting. The top line does not require any special handling, unlike with the scrolling text display.

If a plot extends beyond the visible screen, then it is calculated, but does not needlessly consume programmable characters. This is handy for games, where no special precautions are needed when something ‘drifts out of shot’.

Commands

POINTGRAPHICS

Used to initialise the graphics package, so must be used before any other command

BACKGROUND(pattern,lowestchar)

Optional - used to change the background pattern and the number of characters available for plotting.

VGRA.OV defaults to:-

1. plain black background (GRNDPAT=$0000)
2. characters $60 to $FF are available for plotting (GROUND=$60)

Pattern can take any value 0000-FFFF hex. For instance pattern=$AAFF produces a checkered background.

Lowestchar can take any value 0-FF hex. e.g. Value 0 would give all 255 characters over to graphics, so you would no longer be able to use text on the screen.

RESOURCE

Returns the number of unused programmable characters remaining, available for further plotting. After using command FIXFRAME, RESOURCE would return the to max count of plotting characters. After considerable plotting, that number will drop to zero.

MOVETO(x,y)

Move the current plotting point to absolute position x,y

MOVEBY(x,y)

Move the current plotting point by x,y relative to a new position

POINTAT(x,y)

Plot a single point at absolute position x,y

POINTBY(x,y)

Plot a single point by moving relative to the current point and plotting a single point at the new position

LINETO(x,y)

Plot a line from the currrent position to absolute position x,y

LINEBY(x,y)

Plot a line from the current position to a relative position x,y away

CIRCLE(r)

Plot a circle, radius r, at the current plot position

FIXFRAME

Used after all plots are complete - Set all plotting characters available for the next ‘frame’ of plotting – (using the movie film analogy of successive frames on a piece of a film)

Notes

The POINTBY and LINEBY commands are useful where a shape needs to move around the screen. The shape commands just need to be preceded by a MOVETO command to set the starting position.

Notice that the graphics overlay file VGRA.OV is called once with the POINTGRAPHICS command. All other graphics commands assume this overlay is still in memory, they jump straight into the overlay. This is done for speed although it breaks the Polydos rules a bit. If another overlay is used, be careful to use the POINTGRAPHICS command afterwards before plotting, so that the graphics overlay is back in memory again.

Conclusion

This graphics package was first written back in 1983, whilst I worked at Ferranti. My colleague Doug Seaton and I used to bounce ideas and code back and forth. We sweated the code down – all in the name of plotting speed.

Forty years later, I came across a box of Nascom floppy disks in the attic. Neal Crook offered to read them and a few weeks later, emailed me back all the data. Not bad going stored in that uncontrolled environment – and well done Neal.

I've added a couple of tweaks to ensure Nascom 4 as well as Nascom 1-3 programmable character sets work. The code adjusts itself to suit. Take a look at the examples on the disk image to see how the commands go together.

Hope you find the graphics interesting enough to have a go yourself – Bob Edwards April 2024