(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 437 064 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **90313774.3**

(51) Int. Cl.[5]: **G06F 15/72**

(22) Date of filing: **17.12.90**

(30) Priority: **06.01.90 GB 9000325**

(43) Date of publication of application:
17.07.91 Bulletin 91/29

(84) Designated Contracting States:
DE FR

(71) Applicant: **THE MARCONI COMPANY LIMITED**
The Grove Warren Lane
Stanmore Middlesex HA7 4LY(GB)

(72) Inventor: **Hardiman, Alan**
31 Waltham Close
Fareham, Hampshire P016 8EQ(GB)

(74) Representative: **Cockayne, Gillian et al**
The General Electric Company plc Patent Department GEC Marconi Research Centre
West Hanningfield Road
Great Baddow, Chelmsford Essex CM2 8HN(GB)

(54) Computer graphics system.

(57) A method of displaying graphics on a computer terminal screen comprising:
1) digitising an image to be displayed into a digital information array of predetermined dimensions;
2) sub-dividing said information array into a plurality of identically sized pixel matrices, each corresponding in size to a standard character displayed on said screen;
3) assigning a different predefined standard character code to each pixel matrix in the array;
4) defining the screen display of the standard character code assigned to each pixel matrix according to the digitised information in the respective pixel;
5) displaying the defined characters on the screen in continuous arrangement corresponding to the arrangement of pixels in the array so as to display the image.

EP 0 437 064 A2

COMPUTER GRAPHICS SYSTEM

The present invention relates to a computer graphics system for representing graphical images on a terminal screen and in particular to a system for use on a non-graphics terminal.

The terms "graphics" and "graphical images" are used in this application to indicate images displayed on a screen, or part of a screen, which are not limited in the shapes or arrangement of lines in the image, these being selected freely by a user. While it is possible to display graphics on a terminal screen, current systems are expensive in terms of the amount of memory required and moreover require a specific graphics terminal having appropriate pre-programming. "Non-graphics" or "character" screens can only display characters from a predefined set, each character constituting a predefined pixel matrix on a screen. Hence, there are only a limited number of shapes which can be displayed to provide an image and it is not possible to display any freehand image created by the user. However, such terminals are cheaper than graphics terminals.

The present invention arises from an attempt to provide a graphics representation capability for a non-graphics terminal.

According to the present invention, there is provided a method of displaying graphics on a computer terminal screen comprising:

digitising an image to be displayed into a digital information array or predetermined dimensions;

sub-dividing said information array into a plurality of identically sized pixel matrices, each corresponding in size to a standard character displayed on said screen;

assigning a different predefined standard character code to each pixel matrix in the array;

defining the screen display of the standard character code assigned to each pixel matrix according to the digitised image information in the respective pixel matrix in the array;

displaying the defined characters on the screen in continuous arrangement corresponding to the arrangement of pixels in the array so as to display the image.

Preferably, the predefined character codes comprise the codes for ASCII standard characters, the redefinition of characters typically taking place at the terminal.

The continuous arrangement is conveniently formed such there are no gaps between any pixels.

The coding for the pixels may typically comprise a sub array of information comprising X and Y coordinates of the pixels and indication of the presence or absence of an image line. It is particularly preferred that the digitised information is analysed and points connected by a straight line are defined by the appropriate formula for X and Y coordinates, no information being stored about the points intermediate the end points of said straight line.

The image to be displayed may typically be digitised using a pressure sensitive device, light or heat activated device or other devices such as X-ray devices, pencil cameras, sound devices etc.

A specific example of the present invention will now be described for use on a Digital Equipment Corporation VT220 device in conjunction with a Micro-Vax II computer using Vax Pascal running under the VMS operating system and is intended for the reproduction of personal signatures.

The first step in the process is the digitisation of the signature and this can be done using any conventional image digitiser. One particular preferred method of digitising the signature is using a digitising tablet such as a "sign On" (Alan Liebart Associates). In this case, the signature is written on a tablet which is sampled at 80 times a second and which captures and digitises the data into a window of 2550 by 1300 dots. This information can be transferred via an asynchronous RS232 line into the host computer. The data is received in an encrypted ASCII format which is decodable using a commercially available algorithm and appropriate software and is recorded as series of three variables, namely X-coordinate, Y-coordinate and Z value (the Z value defines whether the pen is in contact with the paper, 0 = on, 1 = off).

Character Definition

The terminal has commonly used character shapes e.g. A, B, % etc. stored permanently within in ROM made up of a combination of dots (often called 'pixels') in a matrix of 8 x 10 (8 horizontal by 10 verticle). There is also an option where user defined character shapes can be downloaded (transferred from the host computer to the terminal) by means of 'escape sequences'.

The example below shows how an exclamation mark character can be re-defined to be an 'A' character. This information was obtained from a VT220 terminal handbook supplied by the manufacturer.

A dot in a character matrix is shown as a '*' and a space is shown as '.'.

```
      12345678
a0    ........
a1    ........
a2    ...**...
a3    ..*..*..
a4    .*....*.
a5    .******.
b0    .*....*.
b1    .*....*.
b2    ........
b3    ........
```

Each column of the character definition is downloaded as 2 numbers (a and b). The value 'a' contains a binary representation of a0 - a5 (6 bits). The value 'b' contains the representation b0 - b3 (4 bits). A value of 64 is added to these binary values and one is subtracted from the final value. The binary representations are obtained by setting a binary bit if a dot (pixel) is present.

Val/Column	Pixels	Binary + 64	Sub.1	Final	ASCII
Val a. col. 1	.....	01000000	= 40H -1 =	3FH =	'?'
Val a. col. 2	**....	01110000	= 70H -1 =	6FH =	'o'
Val a. col. 3	*.*...	01101000	= 68H -1 =	67H =	'g'
Val a. col. 4.	*..*..	01100100	= 64H -1 =	63H =	'c'
Val a. col. 5.	*..*..	01100100	= 64H -1 =	63H =	'c'
Val a. col. 6.	*.*...	01101000	= 68H -1 =	67H =	'g'
Val a. col. 7.	**....	01110000	= 70H -1 =	6FH =	'o'
Val a. col. 8.	.....	01000000	= 40H -1 =	3FH =	'?'
Val b. col. 1.	....	01000000	= 40H -1 =	3FH =	'?'
Val b. col. 2.	..**	01000011	= 43H -1 =	42H =	'B'
Val b. col. 3.	....	01000000	= 40H -1 =	3FH =	'?'
Val b. col. 4.	....	01000000	= 40H -1 =	3FH =	'?'
Val b. col. 5.	....	01000000	= 40H -1 =	3FH =	'?'
Val b. col. 6.	....	01000000	= 40H -1 =	3FH =	'?'
Val b. col. 7.	..**	01000011	= 43H -1 =	42H =	'B'
Val b. col. 8.	....	01000000	= 40H -1 =	3FH =	'?'

The values are then ready for downloading. The escape sequence is structured in such a way that the final values are listed so that the eight 'a' values are output first (col. 8 first down to col. 1) and then the eight 'b' values are output (col. 8 first again down to col. 1).

The format is as follows:

```
DCS Pfn;Pcn;Pe;Pcms;Pw;Pt {[Name][8 a.vals]/[8 b.vals]ST
```

Where:

DCS	Start of escape sequence ESC P.
Pfn	Font buffer number.
;	Delimiter.
Pcn	1st character to define in ASCII char. set.
;	Delimiter.
Pe	Specifies whether to erase all or only new chars.
;	Delimiter.
Pcms	Matrix Cell Size.
;	Delimiter.
Pw	Screen Width.
;	Delimiter.
Pt	Specifies whether font is full cell or text.
{	Signals the end of parameter characters and starts a download function.
[Name]	Name of font.
[8. a. vals]	8 a. values col. 8 down to col. 1.
/	Delimiter between a. and b. values.
[8. b. vals]	8 b. values col. 8 down to col. 1.
ST	String terminator ESC\.

The exclamation mark character '!' will now be an 'A' character but it will be appreciated that similar techniques can be used to re-define any set character or any other shape.

The signature plotting program takes the ASCII X, Y, Z coordinate data from the data file and converts them into integer values. In certain cases all the possible points between two sets of coordinates are calculated. The points are in the range 1 to 1300 in the Y direction and 1 to 2550 in the X direction. In order to display this information it is necessary to define a block of character shapes that, when printed out in the correct order, will represent the signature. The program currently uses a block of 48 characters (3 rows of 16) as its graphics window. This could be adjusted to user requirements although there is a limitation in the number of characters it is possible to define.

Each character uses a matrix of 8 x 10 (10 rows, 8 cols.). A block of 16 x 3 characters gives a resolution of 30 dots (pixels) in the Y direction by 128 in the X. Each point is then scaled to this resolution, and the character that the particular dot will be contained in, is calculated. Once the actual character is known it is necessary to calculate the row and column with the character. In order to set a bit in a character cell, it is necessary to check if the dot (pixel) is in the top six or bottom four rows of the cell as they are stored separately. This type of storage is used as different logical operations need to be performed on the pixel data depending on the row number within the cell. Once all the X, Y points have been scaled down to

the display window size, and all the dots (pixels) have been set in the character definitions, the signature can then be displayed. The characters are downloaded into the terminal by means of 'escape sequences' and the new character set becomes active. The block of characters are printed out on the screen in the correct order. Other 'escape sequences' to position the screen cursor are also used at this point. The signature will now be displayed on the screen. The character set is then set back to the original definitions so that normal characters will be available when the program terminates.

The program operates in the following sequence:

a. Declare and Initialise Data Structures.

b. Read File. The data file that contains the X, Y data in ASCII form is opened for read access. The first X coordinate is obtained by reading characters up to a colon (':') into a string. This string will then be converted to an integer representing that value. e.g. data from file 1234: (5 characters) is converted to an integer value of 1234 (One Thousand Two Hundred and Thirty Four).

The same operation is performed to obtain the Y and Z values for the first point. This will result in all three values being stored as integer variables to represent the current X, Y and Z values respectively.

The integer values are then compared with two values representing the maximum and minimum X and Y values so far read from the file. Of course in the case of the first set of coordinates read, these values will become the new limits.

The Z value is at this point written to a large integer array, containing all the Z values read from the file. The procedure to fill in dots between values is then called. If the current point is not the first point from the file and the previous point's Z value was 0 (representing the pen being on the paper) the dots will be filled in. A detailed explanation of this procedure will be given later.

The above action is then performed on the entire data file until the End-Of-File (EOF) marker is reached. The file is then closed.

The values of maximum and minimum X and Y will now be set correctly as the entire file has been compared.

c. Fill In Dots Between Points. A procedure within the program will (if necessary) fill in all possible dots between two points. the first point will of course have no previous value to join to so no action will be taken. Similarly no action will be taken if the pen was not on the paper for the previous point. This will be known by examining the previous point's Z value. If the value is 1 this will indicate that the pen was not on the paper and that a line should not be drawn between the points. A value of 0 for the previous Z value indicates that the point should be joined.

To find all the points on a line between two points the mathematical equation for a straight line ($y = mx + c$) is used.

The method of this is as follows.

i. Calculate gradient (m) =
(prev. Y val. - cur. Y val.)/(prev X val. - cur. X val.)

ii. Calculate y axis intercept of line (c) =
(prev. Y val. - (gradient x prev. X val.))

By knowing the two values 'm' and 'c' it is possible to calculate every corresponding Y point, by placing all the intermediate X points into the equation. This operation is also performed for calculating every X point by placing all the intermediate Y points into the equation. this is to cater for extreme gradients on both axes. A gradient of 45 degrees will produce identical points for both calculations.

All new calculated values are placed in the large integer array holding the X and Y points. An attempt to save array space is made, where each value that is written to the X and Y integer arrays is checked with the previous value written. If these are the same there is no advantage in storing this point. The point is therefore discarded. This will of course not cater for all duplications of coordinates. The Z values for calculated points are not needed (the pen is known to be on the paper), and are therefore not calculated. Consequently the array of Z points will only have an entry at an index where the point was actually read from the file.

At the end of this procedure the current values of X, Y and Z are made the previous values, for comparison with the next set of coordinates.

d. Find-Scale. This procedure scales the signature so that it will best fit the size of the screen window. The maximum and minimum X and Y values obtained when the data file was read are used here.

e. Calculate Offset Into Screen Window. The offset into the screen window is calculated by taking the X, Y coordinate data from the arrays and adjusting them by subtracting the scaling factor (previously calculated). A value in the range 1 to 128 will be calculated for X, and a value in the range 1 to 30 for Y. The row number is then calculated. A Y value of 1 to 10 indicates top row (row 3), 11 to 20 middle row (row 2) and 21 to 30 bottom row (row 1). The Y dot position into its character cell is calculated by

subtracting 0, 10 or 20 respectively.

The character number is calculated using the X offset (1 - 128) and the row number. The character number will be in the range 1 - 48. Once this is known it is possible to determine the X dot position into the particular character cell, using a modular 8 calculation.

f. Put Dot (Pixel) In Character Definition. This procedure sets a bit corresponding to the X and Y positions in the required character definition. The Y dot position is used to determine the value that should be logically ORed with the existing mask to set a bit. Y values of 1 to 4 are stored in one array and values 5 to 10 in another. This is because of the way in which character shapes are downloaded. This is shown in greater detail earlier in the document. Logically ORing the values means that if a dot (pixel), has been set by a particular value it does not matter if it is set again by subsequent ones. It is quite possible that a number of points from the large scale are going to be represented by the same bit as the 'scale-down' factor is quite large. The values are as follows.

Y dot	Value to be ORed
1	8
2	4
3	2
4	1
5	32
6	16
7	8
8	4
9	2
10	1

g. Plot Signature On Screen. When all the X, Y data has been processed it is necessary to plot the signature on the VT220 terminal screen. This procedure makes extensive use of the 'escape sequences' associated with the VT220 terminal device, and it may be beneficial to study a copy of the handbook.

Initially the new character set is given a 2 character i.d. and the character set is designated by means of an escape sequence. The program uses the i.d. '@' (space and at symbol). The designation is performed by sending the following:

ESC (sp @ where sp = space character.

The font is then downloaded using the escape sequence described in the section on 'Character definitions'. The two arrays of dot (pixel) positions represent the a. and b. values above. The characters 31 through to 79 are defined. All others have no values. The newly defined characters have no meaning unless they are printed out in the correct order. They need to be printed out as 3 rows of sixteen characters directly above each other.

Further escape sequences are used to position the screen cursor to ensure that the rows will be printed out above each other.

Finally it is necessary to reset the screen character set back to its original value to reset the terminal for future operations.

Claims

1. A method of displaying graphics on a computer terminal screen comprising:
   1) digitising an image to be displayed into a digital information array of predetermined dimensions;
   2) sub-dividing said information array into a plurality of identically sized pixel matrices, each corresponding in size to a standard character displayed on said screen;
   3) assigning a different predefined standard character code to each pixel matrix in the array;
   4) defining the screen display of the standard character code assigned to each pixel matrix according to the digitised information in the respective pixel matrix in the array;
   5) displaying the defined characters on the screen in continuous arrangement corresponding to the arrangement of pixels in the array so as to display the image.

2. A method as claimed in claim 1 wherein the image isdigitised using a pressure, light, x-ray, heat or sound activated device or a pencil camera.

3. A method as claimed in claim 1 or 2 wherein the predefined character codes are codes for ASCII standard characters.

4. A method as claimed in any preceding claim wherein the pixels forming the array are displayed without any gaps therebetween.

5. A method as claimed in any preceding claim wherein the coding for the pixzels comprises a sub array of information comprising X and Y coordinates of the pixel and an indication of the presence or absence of an image line.

6. A method as claimed in any preceding claim wherein redefinition of characters takes place at the terminal.

7. A method as claimed in any preceding claim wherein the digitised information is analysed and points connected by a straight line are defined by the appropriate formula for X and Y coordinates, no information being stored about points intermediate the end points of said straight line.