

REPLACED SHEETS		SG
☒	Description : paged1..... to paged8.....	
☒	Claim(s) : paged9..... to paged10.....	
☒	Drawing(s) : paged1..... to paged9.....	
☒	Abstract : paged11..... to paged-.....	
DATE OF RECEIPT		

GRAPHICS SYSTEM USING QUADRATIC POLYNOMIAL FRAGMENTS

The present invention relates to object graphics and, in particular, discloses a means by which objects can be described, and manipulated by dividing each object into a number of fragments and mathematically describing each fragment by a quadratic polynomial..

Most object graphics systems use a frame store to hold a pixel based image of the page or screen. The outlines of individual objects are calculated, filled and written into the frame store. For two dimensional graphics, objects which appear in front of other objects are simply written into the frame store after the background object, thereby replacing the background on a pixel by pixel basis. Such systems are typically expensive in terms of hardware and the time taken to create a full page image.

Real time image generation for raster displays can be achieved by calculating the image in raster order. This means that each scan line must be calculated as it is encountered. This requires that the intersection points of each scan line with each object outline are calculated, and then filled.

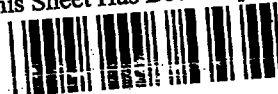
Traditionally, the calculation of intersections of outlines are performed using straight line or spline formats. The most popular spline formats are various forms of parametric cubic polynomials, such as Bezier curves. If cubic Bezier splines are used, intersection calculation involves calculating the roots of a cubic polynomial at a particular scan line to obtain the parametric positions of the intersections, and substituting these into the polynomial to obtain the actual pixel intersections. Whilst it is possible to build hardware to perform this function in real time, such hardware is highly complex, and not suited for consumer priced equipment.

It is an object of the present invention to provide an alternative to the above through provision of a simplified format for representing object outline data which can be readily and quickly manipulated within an object based graphics system.

In accordance with one aspect of the present invention there is disclosed an object based graphics system characterised by the representation of objects by quadratic polynomials.

In accordance with another aspect of the present invention there is disclosed an object based graphics system in which individual objects are represented by a plurality of curve fragments in which each curve fragment is defined by a quadratic polynomial.

In accordance with a further aspect of the present invention there is disclosed an object based graphics system wherein each object is divided into a plurality of curve fragments each being defined by a START_PIXEL location, a Δ PIXEL value, a $\Delta\Delta$ PIXEL



12000000

value, a START_LINE, and an END_LINE wherein the pixel location of said curve fragment on any line_(n) of the displayed image is given by:

$$\text{PIXEL}(\text{line}_{n+1}) = \text{PIXEL}(\text{line}_n) + \Delta\text{PIXEL}(\text{line}_n)$$

$$\Delta\text{PIXEL}(\text{line}_{n+1}) = \Delta\text{PIXEL}(\text{line}_n) + \Delta\Delta\text{PIXEL}$$

where

$$\text{PIXEL}(\text{line}_n = \text{START_LINE}) = \text{START_PIXEL}; \text{ and}$$

$$\Delta\text{PIXEL}(\text{line}_n = \text{START_LINE}) = \Delta\text{PIXEL}.$$

In accordance with a further aspect of the present invention there is disclosed an object based graphics system characterised in that each intersection between an object outline and a scan line can be calculated by two mathematical operations, when the previous intersection is known. Preferably, the mathematical operations are additions.

Embodiments of the present invention will now be described with reference to the drawings in which:

Fig. 1 illustrates a single quadratic polynomial fragment (QPF);

Fig. 2 illustrates an object formed by two QPFs;

Fig. 3 illustrates an object formed by four QPFs;

Fig. 4 illustrates an object formed by six QPFs;

Fig. 5 illustrates an object formed by five QPFs;

Fig. 6 illustrates an object formed by eight QPFs;

Fig. 7 illustrates another object formed by eight QPFs;

Fig. 8 is a schematic block diagram representation of an object based graphic system which utilises QPFs;

Fig. 9 is a data flow diagram of the real time object (RTO) processor of Fig. 8 which indicates those operations occurring on the QPFs,

Fig. 10 shows an object list and the configuration of QPFs for each object in the list;

Figs. 11 and 12 show the object and QPF data word formats of the list of Fig. 10;

Fig. 13 illustrates the organisation of the QPF memory of Figs. 8 and 9;

Fig. 14 shows the data format of a single QPF in the QPF memory of Figs. 8 and 9;

and.

Figs. 15 to 19 shows examples of QPF formation for a single object.

Referring to Fig. 1, a quadratic polynomial fragment QPF which forms part of an object outline is illustrated extending from a start line to an end line within a raster display. The QPF is a curve characterised by a constant change in slope of the line and in this manner, each fragment of any object outline can be described using a quadratic polynomial.

This Sheet Has Been Replaced



178800

Because each QPF changes slope in a constant manner along its length, each QPF can be readily defined by a number of parameters.

The first of these parameters is the START_LINE which specifies the first line in raster order upon which the QPF appears.

The next parameter is the START_PIXEL which specifies the pixel value (i.e. location) of the QPF on the start line.

The next parameter is Δ PIXEL which specifies the slope of the line. Δ PIXEL is added to start pixel for each raster line of the display.

The next parameter is $\Delta\Delta$ PIXEL which specifies the curvature of the line. $\Delta\Delta$ PIXEL is added to Δ PIXEL each raster line.

The last parameter is the END_LINE which specifies the last line on which the QPF appears.

With the foregoing parameters, the pixel locations on each of the scan (raster) lines of the QPF can be determined in the following manner:

$$\text{PIXEL}(\text{line}_{n+1}) = \text{PIXEL}(\text{line}_n) + \Delta\text{PIXEL}(\text{line}_n)$$

$$\Delta\text{PIXEL}(\text{line}_{n+1}) = \Delta\text{PIXEL}(\text{line}_n) + \Delta\Delta\text{PIXEL}$$

where

$$\text{PIXEL}(\text{line}_n = \text{START_LINE}) = \text{START_PIXEL}; \text{ and}$$

$$\Delta\text{PIXEL}(\text{line}_n = \text{START_LINE}) = \Delta\text{PIXEL}.$$

Turning now to Figs. 2-7, there are shown a number of individual objects each of which being defined by a plurality of QPF's.

In Fig. 2, a square object is shown defined by two QPFs which define the left and right sides of the square respectively. Each QPF commences on a common start line and ends on a common end line. In this manner, the QPFs of the square define the boundaries at which particular pixel value (such as a colour) can be asserted so as to display the square.

In Fig. 3, a lozenge-shaped object is shown which is defined by four QPFs in which the end line for one QPF represents the start line for the next, adjacent QPF.

It is apparent from Figs. 2 and 3, that it is not possible for a QPF to be aligned with a scan line which would imply infinite slope in, for example, the embodiment shown in Fig. 1.

Fig. 4 shows an alternative object formed by six QPFs. Those QPFs that appear vertical (perpendicular to the scan lines) have $\Delta\text{PIXEL} = 0$ and $\Delta\Delta\text{PIXEL} = 0$. Where a QPF has a constant (linear) slope, ΔPIXEL is non-zero and $\Delta\Delta\text{PIXEL}$ equals 0.

The QPF format described herein has specific application in real time object graphics and is useful in a system such as that disclosed in Australian Patent Application



No. (Attorney Ref:(RTO7)(202788)) entitled "A Real Time Object Based Graphic System", claiming priority from Australian Patent Application PL2147 of 29 April 1992 and the disclosure of which and the equivalent European application is hereby incorporated by cross reference.

Turning now to Fig. 8, such a real time object (RTO) graphics system is shown which includes a controlling processor 11 connected via a processor bus 17 to a processor memory 12 which includes ROM and RAM. The processor 11 operates to form a page display list of objects which are used to form an image. The page display list is stored in the RAM portion of the processor memory 12 in which the objects can be derived from the ROM portion of the processor memory 12 or alternatively input to the system 10 via any known external means such as a disc drive, a digitising tablet of a graphics work station or the like.

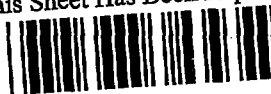
Connected to the processor bus 17 is a RTO processor 13 which manipulates QPFs of objects in the page display list so as to output rasterised image data to a colour lookup table 15 for reproduction on a display 16 such as a video display unit or printer. The RTO processor 13 also connects via a QPF bus 18 to a QPF memory 14 formed of static RAM which is used for the storage of QPFs whilst the processor 13 is calculating the image data.

Accordingly, because the processor 13 can only accept QPFs, the objects which form the page display list produced by the processor 11 must also be described as QPFs.

Where the processor 11 and preformatted objects are configured using cubic polynomials such as Bezier splines, these polynomials can be converted to QPFs in the manner described in Australian Patent Application No. (Attorney Ref: (RTO9)(203174)) entitled "Bezier Spline to Quadratic Polynomial Fragment Conversion", claiming priority from Australian Patent Application PL2149 of 29 April 1992 and the disclosure of which is hereby incorporated by cross reference.

Referring now to Fig. 9, the data flow diagram of the RTO processor 13 shows an image fetch unit 20 connected to the processor memory 12 via the bus 17. Objects and their QPFs are fetched from the memory 12 by the unit 20 and are output to a QPF first-in-first-out (FIFO) register 21 which is four words deep. Data in the FIFO 21 is tagged as object or QPF data as per the formats of Figs. 11 and 12 to be later described. The FIFO 21 is used to decouple fetching from data processing, so as to increase the access efficiency to the processor bus 17 and the calculated efficiency.

This Sheet Has Been Replaced





Data is output from the FIFO 21 into a preprocessing pipeline 22 which performs a series of calculations on the data before it is stored in the QPF RAM 14. These operations are:

- (23) applying scaling and translation factors for the current object to each QPF;
- (24) filtering out QPF's which will not effect the display;
- (25) iteratively recalculating the values in a QPF which start before the first line of the display, to yield a QPF starting on the first line of the display; and
- (26) applying a correction to the QPF's rendering if the image is to be interlaced (as, for example, on a television display).

From the preprocessing pipeline 22, the QPFs are stored in the QPF memory 14. The QPF data is stored as a series of linked lists, one for each line in the image to be formed. After all of the QPFs in the image have been fetched and stored in the memory 14, the linked list for each line is sorted by a sorter 28 in order of pixel value (i.e. position), in preparation for rendering of the image to the display 16.

In video applications, image preparation of the new image occurs in one half of a double buffered QPF memory, whilst the other half is used for rendering. However, the QPF memory 18 is single ported, so the image preparation and image rendering sections of the processor 13 must compete for access to the bus 18.

Image rendering is commenced in synchronization with the display 16. Rendering consists of calculating the intersection of QPFs with each line on display 16 in turn, where these intersections define the edges of objects. Each edge (or intersection) is used in the calculation of the level which is to be displayed at a particular pixel position on the scan line.

For each line in the image, the render process steps through the list of QPFs for that line, executing the following steps:

- copy the pixel value (location), pixel level (colour) and effects information into a pixel FIFO 30 to be picked up for fill generation and calculation;
- calculate the values of the QPFs intersection for the next line, or discard QPFs which terminate on the next line; and
- merge the recalculated QPF into the list of QPF's starting on the next line.

Rendering and recalculation has the highest priority on the QPF bus 18, however the bus 18 is freed from storage or sorting access whenever the pixel FIFO 30 is filled, or all the QPFs for the current line have been processed.

Data is sequenced out of the pixel FIFO 30, which is sixteen words deep, under the control of a pixel counter which is incremented by each pixel clock cycle derived from the

display 16. The fill generator 31 resolves the priority levels of the objects in the display, and outputs the highest visible level at each pixel position to the colour lookup table 15 prior to display. The fill generation can be modified by an effects block 32, so as to implement visual effects such as transparency.

The organisation of the data making up an image in the processor memory 12 is shown in Fig. 10. This shows one object list, although there may be many such lists, each in the format shown.

In the preferred embodiment of the graphics system 10, the objects in the list are placed contiguously in memory, with each object (Object 1, Object 2, ...) occupying nine 16 bit locations. The last two words of each object are *QList(low) and *QList(high) which form a pointer to the start of the QPF list for that particular object. The QPF list can reside anywhere in the memory 12. The remainder of the object description is made up of scaling and translation factors (LScale, PScale, Δ PScale, $\Delta\Delta$ PScale, LOffset and POffset), the object's level (6 bits) and effects (2 bits) (Level), and an eight-bit object tag (Tag)

LOffset represents a palette of colours from which Level selects the actual colour of the object. In this manner, Level and LOffset can address 24 bit colour data permitting over 16 million colours to be displayed using the system 10. Although the RTO processor 13 only manipulates Level, the colour look-up table 15 also has access to the processor bus 17 whereby the processor 11 can adjust the particular palette. By writing to the colour look-up table 15 to display the appropriate colour for that object.

POffset represents a pixel translation amount (X translation for video display)

LOffset represents a line translation amount (Y translation for video).

LScale represents a line scaling factor (related to Y size).

PScale represents a pixel scaling factor (related to X size).

Δ PScale represents a pre-calculated factor for scaling Δ PIXEL to give an equivalent X size as PScale.

$\Delta\Delta$ PScale represents a pre-calculated factor for scaling $\Delta\Delta$ PIXEL to give equivalent X size as PScale.

For each object, the QPFs are listed contiguously in memory, each QPF being made up of five 16 bit words. These are, as described earlier, (in order) the START_LINE, the END_LINE, the START_PIXEL, the Δ PIXEL, and the $\Delta\Delta$ PIXEL. Both object and QPF lists in the processor memory 12 must contain at least one item, as there is no provision for a null list.

This Sheet Has Been Replaced



The data types of the different object and QPF components are indicated and their precise formats are shown in Figs. 11 and 12. In addition to the image data, some of the object and QPF components contain check bits and termination bits.

Check bits occupy the top bit of the four scale factors (LScale, PScale, Δ PScale, $\Delta\Delta$ PScale) in an object description and the START_LINE component of a QPF description. In the object list, the check bits must be set to zero. In the START_LINE field, the check bit must be the same as the sign bit. If any of these bits are incorrect, the RTO processor 13 will detect this and abort the image generation.

The bottom bit of the *QList(low) word of each object is used as a termination marker for the object list. This bit is set to a 1 for the final object in the list only. The top bit of the end line word of each QPF is used as a termination marker for the QPF list. For all QPFs except the last, it must be the same as the end line sign bit. For the last QPF, it must be the opposite of the sign bit.

The formatting provided by Figs. 10, 11 and 12 permit the RTO processor 13 to successfully read QPFs for an image and to manipulate those within the preprocessing pipeline 22. After preprocessing, it is necessary for the QPFs to be stored in the memory 14, by which access is obtained using the memory interface 27 of Fig. 9.

The QPF memory 14 is organised in the manner shown in Fig. 13. Fig. 13 shows one bank 33 of QPF memory 14 and if more than one bank is used, each bank is organised in an identical fashion. The bank 33 is divided into four areas comprising a line table 34, and pixel table 35, a bit table 36, and a QPF area 37. The line table 34, which contains one word for each line of the display, is used for storing the start of line pointers and always occupies the bottom of the QPF memory 14. The pixel table 35 is a temporary storage area used for the sorting of pixels carried out by the sort unit 28. The base location is set by a register PTBA, and the size is one word for each pixel displayed in a line, plus two additional words. The bit table 36 is another temporary storage area used in the pixel sort. Base location is set by a register PTBA, and the size of the bit table 36 is the size of the pixel table 35 divided by 32, and rounded up to the nearest whole word. The QPF area 37 contains all of the QPF data. This always occupies the top of the QPF memory 14 in which the bottom address is set by a register QRBA, while the top address, which is also the top address of the bank 33, is set by a register MBTA.

The format of the QPF components stored in the memory bank 33 are indicated in Fig. 14. In the preprocessing pipeline 22, the object scale and offset factors have been applied to each QPF, resulting in PIXEL, Δ PIXEL and $\Delta\Delta$ PIXEL values having greater range and/or precision compared to their original formats in the processor memory 12.



Also, the object level, effects and tag bits have been merged in and stored with each QPF. The start line field need no longer be stored with the QPF, as the QPF lists are based on the start line. There are no check bits, and the end of each linked list is denoted by a zero pointer.

This structure is seen in Fig. 13 wherein the QPF area 37 there are shown two linked lists for lines zero and one of the image. The linked list for line zero commences with a QPF 40 which points to a QPF 41 which in turn points to a QPF 42 at the end of the list. The linked list for line 1 of the image commences with a QPF 43 which points to a terminal QPF 44.

The actual formation and calculation of QPF data can now be described by example using a simple object.

Fig. 15 shows an object of letter "B" shape. This object can be represented by QPF's in the manner shown in Fig. 16.

Fig. 17 shows a Bezier object for the letter "B" which can be transformed to a QPF object using the method disclosed in Australian Patent Application No. PL2149 (Attorney Ref: (RTO9)(203174)).

In Fig. 18, a specific Bezier curve of the object of Fig. 17 is shown, in 4-point format. This is transformed into two QPF's as shown in Fig. 19.

Modifications of the embodiments of the present invention can be made thereto without departing from the scope of the present invention.

This Sheet Has Been Replaced





CLAIMS:

1. An object based graphics system characterised by the representation of objects by quadratic polynomials.
2. An object based graphics system in which individual objects are represented by a plurality of curve fragments characterised in that each curve fragment is defined by a quadratic polynomial.
3. An object based graphics system wherein each object is divided into a plurality of curve fragments characterised in that each said fragment is defined by a START_PIXEL location, a Δ PIXEL value, a $\Delta\Delta$ PIXEL value, a START_LINE, and an END_LINE wherein the pixel location of said curve fragment on any line_(n) of the displayed image is given by:

$$\text{PIXEL}(\text{line}_{n+1}) = \text{PIXEL}(\text{line}_n) + \Delta\text{PIXEL}(\text{line}_n)$$

$$\Delta\text{PIXEL}(\text{line}_{n+1}) = \Delta\text{PIXEL}(\text{line}_n) + \Delta\Delta\text{PIXEL}$$

where

$$\text{PIXEL}(\text{line}_n = \text{START_LINE}) = \text{START_PIXEL}; \text{ and}$$

$$\Delta\text{PIXEL}(\text{line}_n = \text{START_LINE}) = \Delta\text{PIXEL}.$$

4. An object based graphics system characterised in that each intersection between an object outline and a scan line can be calculated by two mathematical operations, when the previous intersection is known.
5. An object based graphics system as claimed in claim 4 characterised in that said mathematical operations are additions.
6. An object based graphics system in which individual objects are represented by a plurality of curve fragments characterised in that each curve fragment is defined by a quadratic polynomial and is derived from a corresponding spline fragment.
7. An object based graphics system as claimed in claim 2 characterised in that each object is further represented by scaling or translation factors.
8. An object based graphics system as claimed in claim 2 characterised in that each object is further represented by colour level and offset information.
9. A method for representing objects in an object based graphics system, said method characterised by representing said objects by quadratic polynomials.
10. A method for representing objects in an object based graphics system, characterised by representing individual objects by a plurality of curve fragments and defining each curve fragment by means of a quadratic polynomial.
11. A method of generating image data wherein features of the image are fragmented and represented using quadratic polynomials.



12. A signal or recording comprising image data in which features of the image are fragmented and represented using quadratic polynomials.
13. Apparatus for converting cubic curve fragments to quadratic polynomial fragments.



This Sheet Has Been Replaced

