



METHOD OF AND SYSTEM FOR DETECTING AND RENDERING OF GRAPHIC ELEMENTS

I. BACKGROUND OF THE INVENTION

5 The present invention relates generally to the field of a rendering engine architecture for computer graphic displays and displays information processing technology. More specifically, the present invention relates generally to a method of and system for a rendering engine architecture wherein graphics or other objects are detected and rendered for display to a user independent of the source of the graphics to be rendered.

10 The two most common forms of display systems are stroke display systems and raster display systems. Stroke display systems include stroke deflection processors and Cathode Ray Tube (CRT) stroke-type displays. Stroke display systems are capable of high-quality symbology with inherent anti-aliasing, due to the ability to position the electron beam in very fine
15 increments and the Gaussian distribution of electrons within the beam. However, stroke display systems are now virtually obsolete and are plagued with problems. For example, CRT stroke-type displays experience high failure rates, high costs, low supply, and diminishing number of suppliers.

Raster display systems include raster image processors and raster-type
20 CRT or flat panel displays. Raster display images are typically more quantized than those of stroke images, either in terms of the number of horizontal lines in the image, or in the case of mosaic displays such as LCDs, the pixel resolution. Raster display images often require special processing techniques, such as anti-aliasing, to approach the symbol quality level available in legacy stroke
25 systems. Raster-mode flat panel displays exhibit lower failure rates, lower costs, are readily available and represent the state of the art.

Flat panel displays can not directly replace CRT stroke-type displays in a stroke display system, however, because they require a raster image source. This suggests that when a CRT stroke-type display is replaced with a state of
30 the art flat panel display, the corresponding stroke deflection processor must also be replaced. Therefore, a method of and system for replacing stroke-type display with a flat panel display is desired which does not require the corresponding stroke deflection processor to be replaced. Eliminating the need



to replace a stroke deflection processor reduces the costs and risks associated with a stroke-type display replacement.

5 The translation of information from its input domain to the graphics domain is necessarily destructive in a system that provides only graphics as output. Graphics are rendered with the intent of satisfying a specific set of display requirements. When display requirements change but the rendering method cannot change for the input graphics, one is forced to compromise graphics quality with conventional systems.

10 For example, a bit map image can include graphics that have been rendered for a specific display resolution and size. When the bit map image is to be rendered on a display having a different resolution and size than the original display, the bit map is converted using bilinear interpolation under conventional methods. The process can also involve converting stroke information to a bit map image (e.g., by sampling deflection, intensity, color, etc.) and then processing the image with software. Information is distorted as well as lost during this process, yielding a bit map which indeed can be rendered on the new display (e.g., flat panel displays) but with lower quality than if the bit map had been rendered directly for the new display. Further, the conversion process does not handle overlapping symbols and is not performed in real-time.

20 For a hybrid stroke/raster display system using a flat panel display, a conventional method includes over-sampling stroke deflection, color and intensity information from a stroke deflection processor in order to create a stroke symbology bit map with color and intensity for each pixel, then processing the stroke symbology bit map and merging the processed stroke symbology bit map with digitized raster video from a raster image processor. The merged stroke/raster images are then provided to a flat panel display. Stroke symbology bit map processing can include anti-aliasing, edge detection, image smoothing, and contrast enhancement. The possibility of errors increases with this method due to the longer vectors and larger symbols, when vectors and symbology intersect, when symbology of differing color are in close proximity, and where symbology is comprised of small features of complex shapes or arcs. The net effect of conventional methods is to generally blur stroke symbology, thus creating wider than necessary symbology with less

25

30

accuracy than desired. This becomes a problem especially for small symbols, for symbols which must be rendered accurately, and where symbology elements intersect or overlap.

FIG. 1, which represents stroke symbology image digitized into a bit map before image processing, illustrates why this is a problem for some conventional approaches. The image-processing component in conventional approaches must process this bit map without the benefit of information lost when symbols intersect or overlap. The reference numeral 2 represents the two green lines connecting two small circles; the reference numeral 4 represents a red circle in the upper left corner of the figure; and the reference numeral 6 represents a blue line (extending from the upper left corner to the lower right corner). When conventional processing methods are applied for intersecting lines, the intersection appears on a display to include a dot with diameter larger than the width of either line. Small circles may also appear as dots when processed using conventional methods. Thus symbology becomes ambiguous. When conventional processing methods are applied where symbology of differing colors are in close proximity, a third and false color is perceived on a display, creating still more ambiguity. Further, where symbology of differing colors overlap, only one of the symbols is perceived on a display, effecting a loss of information from the other symbol.

Another example involves a method where vector graphics are rendered into a bit map image intended for a flat panel display. When viewed on the display, the graphics contain artifacts such as aliasing (for example, stair stepping of lines drawn at angles other than 0 degrees or 90 degrees). Conventional systems include methods for post-processing the bit map image using techniques which detect lines and edges, then alter the bit map by blurring the detected lines and edges. When viewed on a flat panel display, the effects of aliasing appear diminished, but at the expense of graphics resolution and quality.

30

II. BRIEF SUMMARY OF THE INVENTION

The following summary of the invention is provided to facilitate an understanding of some of the innovative components unique to the present invention, and is not intended to be a full description. A full appreciation of the

various aspects of the invention can be gained by taking the entire specification, claims, drawings, and abstract as a whole.

In accordance with the present invention, there is provided a method for detecting graphic objects (e.g., vectors) that have been created for a first display (e.g., stroke-type display), the graphic objects representing an image, and re-rendering the graphic objects in a form adapted for a different display (e.g., raster-type flat panel display). The method includes the steps of: receiving formatted graphics from a source, e.g., video, that have been formatted for the first display; detecting a plurality of graphic objects within the formatted graphics; creating a graphics array that represents the original image, which contains information in a predetermined, prioritized manner; and re-formatting or re-rendering the plurality of graphics objects for the different type of display. The re-rendered graphics objects are then stored for later use or displayed on the different type of display. The objects within the graphics array are then manipulated by anti-aliasing in order to smooth lines, texturing to enhance the appearance of graphics objects, alpha blending to combine graphics objects and maintain correct color perception, haloing to increase contrast between graphics objects and the background scene, interpolation for scaling and smoothing graphics objects, fogging to provide the perception of depth, fills in order to color graphics objects, merging with other graphics information and sources, and other functions. The step of detecting includes the step of detecting start and end points of vectors present in the graphics objects along with color and intensity. Also, the length, direction, color, and draw rate of each individual vector within the graphics objects are detected.

In accordance with another aspect of the present invention, there is provided an apparatus for detecting graphic objects that have been created for a first display, the graphic objects representing an image, and re-rendering the graphic objects in a form adapted for a different display. The apparatus includes means for receiving formatted graphics from a source, the formatted graphics having formatted for the first display; detecting means, in communication with said receiving means, for detecting a plurality of graphic objects within the formatted graphics; generation means, in communication with said detecting means, for creating a graphics array from the plurality of graphics objects that represents the image based on the formatted graphics;

and re-rendering means, in communication with said generation means, for re-rendering the plurality of graphics objects formatted for the different display. The apparatus further includes means to digitize the graphics objects.

5 In an alternate embodiment, the present invention also combines the re-rendered graphics objects with other information including imagery, video, or graphics from another source in order to provide a more comprehensive display format to view.

10 The novel components of the present invention will become apparent to those of skill in the art upon examination of the following detailed description of the invention or can be learned by practice of the present invention. It should be understood, however, that the detailed description of the invention and the specific examples presented, while indicating certain embodiments of the present invention, are provided for illustration purposes only because various changes and modifications within the spirit and scope of the invention will
15 become apparent to those of skill in the art from the detailed description of the invention and claims that follow.

III. BRIEF DESCRIPTION OF THE DRAWINGS

20 The accompanying figures, in which like reference numerals refer to identical or functionally-similar elements throughout the separate views and which are incorporated in and form part of the specification, further illustrate the present invention and, together with the detailed description of the invention, serve to explain the principles of the present invention.

25 FIG. 1 (prior art) is a bit map illustrating colors, small circles, intersections, and overlap.

FIG. 2 is a block diagram illustration of an embodiment 10 of the overall system components in accordance with the present invention.

FIG. 3 is a block diagram illustration of the Graphics Detection Processor 200 in accordance with the present invention.

30 FIG. 4 is a block diagram illustration of the Encode Processor 300 in accordance with the present invention.

FIG. 5 is a block diagram illustration of the Graphics Rendering Processor 400 in accordance with the present invention.

FIG. 6 is a block diagram illustration of the Display 600 in accordance with the present invention.

FIG. 7 is a block diagram illustration of an alternate embodiment 800 of the information detection and regeneration process in accordance with the present invention.

FIG. 8 is a block diagram illustration of a more specific implementation 20 of the alternate embodiment 800 of the overall system components in accordance with the present invention.

FIG. 9 is a block diagram illustration of the Graphics Rendering Processor 1000 in accordance with the alternate embodiment 20 of the present invention.

FIG. 10 is a block diagram illustration of the Display Formatter 1100 in accordance with the alternate embodiment 20 of the present invention.

FIG. 11 is a diagram illustration of the Merge Processor 500 in accordance with the alternate embodiment 20 of the present invention.

FIG. 12 is a block diagram illustration of an implementation of the alternate embodiment 20 of the present invention to solve the problem of displaying high-quality, anti-aliased color stroke symbology along with high-quality color raster video on a color flat panel display

FIG. 13 is a block diagram illustration of the stroke vector detection function of the alternate embodiment of the present invention.

FIG. 14 is a block diagram illustration of the stroke vector rendering function of the alternate embodiment of the present invention.

FIG. 15 is a block diagram illustration of the raster formatter function of the alternate embodiment of the present invention.

IV. DETAILED DESCRIPTION OF THE INVENTION

In order to facilitate an understanding of the present invention, a brief discussion is provided wherein a conventional implementation to stroke to raster conversion is compared with an embodiment of the present invention. The approaches between a conventional implementation and that of the present invention vary considerably. The functions necessary for converting analog stroke symbology inputs to high quality, anti-aliased symbology with

raster video on a color flat panel display are presented with the manner in which implemented conventionally versus the present invention.

Stroke deflections and video are digitized through over-sampling. Instead of creating a bit map as in conventional approaches, the present invention detects start and end points for individual vectors within specific symbols, along with their color and intensity. Individual vectors are distinguished through a change in direction, a change in color, a change in intensity, a change in draw rate, or by the assertion or de-assertion of the symbology blanking signal. This effectively recreates the original vector display list used by the original stroke deflection processor. Once the original display list is recreated, stroke symbology can be re-rendered without the errors experienced in conventional approaches. This is because the vectors from the display list are then rendered by graphics rendering components using state-of-the-art algorithms for anti-aliasing, alpha blending, and other graphics rendering functions. These graphics rendering components are the same as those used on graphics cards for personal computers and workstations, and implement algorithms well understood by those skilled in the art. The process of dynamically shutting off and then turning on a stroke vector is called occlusion. Occlusion is typically accomplished for a stroke display system by setting the stroke video signal for zero beam intensity while the deflection is in the occlusion area. For the present invention, occlusion areas are precisely maintained as a result of re-rendering only those vectors present simultaneously with stroke video. The smallest symbols as well as intersections and arcs are accurately rendered with the highest possible quality. Once the stroke symbology has been re-rendered in raster form, it can be merged with digitized raster video.

Both conventional approaches and the present invention over sample the stroke video inputs for analog to digital conversion. Both conventional approaches and the present invention perform a merge of stroke symbology with digitized raster video in similar manners. However, conventional approaches have the problem of performing bit map or image processing in the presence of arcs and intersections of varying colors to a degree which typically has been performed in non-real time systems or in systems with parallel digital signal processing components.

The present invention shifts the problem from that of processing bit maps in real time (see discussion above with respect to FIG. 1) to one of detecting the beginning and end of individual vectors, which can be performed in real time. The present invention removes the effects of lost and distorted stroke symbology information, and re-renders the stroke symbols in a manner optimized for a high resolution, color flat panel display. Unlike conventional approaches, which must process a bit map, the present invention processes a display list of individual vectors. The display list represents each and every vector rendered for the stroke symbology with no loss of information.

Conventionally, analog stroke video A/D is performed by oversampling relative to write rate and display pixel resolution. Digitized stroke video data processing is performed by first creating a bit map. Rendering of symbology with anti-aliasing is performed by post processing the bit map using edge detection, image smoothing, contrast enhancement, nearest neighbor analysis (pseudo anti-aliasing) and other graphics and image processing techniques well understood by those skilled in the art. The analog raster video A/D is performed by oversampling relative to the input resolution and display pixel resolution. The stroke symbology and raster video are merged through digital summation with symbology precedence or priority. The end-to-end effect of this conventional approach is that the quality of the symbology is compromised while not fully realizing the capabilities of a high-resolution color, flat panel display.

By comparison, in one embodiment of the present invention described herein, analog stroke deflection and video A/D is performed by over-sampling relative to write rate and display pixel resolution. Digitized stroke video data processing is performed by creating a display list. Rendering of symbology with anti-aliasing and alpha blending is performed by graphics rendering of simple vectors (true anti-aliasing) with sub-pixel positioning of the end points, using algorithms using anti-aliasing and alpha blending algorithms well known and understood by those skilled in the art. Anti-aliasing algorithms have the capability to modulate pixels along and around a vector in a manner that makes the line appear straight and smooth when viewed on a display. This is particularly important for flat panel displays which would otherwise present a vector as a stair-stepped line. Alpha blending enables overlapping vectors to

be rendered in a manner which ensures that the highest priority vector is not obscured by lower priority vectors. Alpha blending algorithms modify pixel colors along and around vectors in order to ensure that vectors of differing colors that are rendered in close proximity or in an overlapping manner do not present a third (false) color when viewed. The analog raster video A/D is performed by oversampling relative to input resolution and display pixel resolution. The stroke symbology and raster video are merged through digital summation with symbology precedence. The end-to-end effect of this approach of the present invention is that maximum quality of symbology is realized using fullest capabilities of a high-resolution, color flat panel display.

Having described the basic operation of an embodiment of the present invention, attention is now turned to FIG. 2 where there is shown an embodiment 10 of the present invention represented by its major components. Each component (200 through 600) is further described herein. For the present invention, graphics objects includes all information that is present in a graphics display system. The present invention enables one to render graphics for display purposes in an optimal manner without regard for the source of the graphics to be rendered. In particular, graphics already rendered in a manner that may not be the desired method can be re-rendered using the desired method. The present invention can convert analog stroke symbology inputs to high-quality, anti-aliased symbology with background raster video, for example, display on a color flat panel display.

The particular values and configurations discussed herein can be varied and are cited merely to illustrate particular embodiments of the present invention and are not intended to limit the scope of the invention. In the following discussion of this embodiment, it is desirable to process analog color video that has been rendered as stroke (vector) video signals for a vector-type display in order to render the video in an optimized manner on a raster, color flat panel display. A restriction exists in such a system wherein the source of the stroke (vector) video cannot be altered. Although the following description has been provided with a stroke to raster graphics rendering, it will be apparent to those skilled in the art that the present invention has other applications, including, without limitation, image enhancement, video scene analysis,

character recognition, target recognition, and other forms of information translation and analysis.

Referring again to FIG. 2, the Graphics Detection Processor 200 detects the individual vectors rendered by the original source, such as stroke (vector) video input for each video frame. To accomplish this, the Graphics Detection Processor 200 first digitizes the horizontal deflection (x display position) and vertical deflection (y display position) signals, color signals, intensity signals, symbology blanking signal and other signals from the stroke (vector) video using analog to digital converters (A/D) to form a digital sample of a vector in a manner well known to those skilled in the art.

Then the direction of a vector, $((x_2 - x_1), (y_2 - y_1))$, where x_1 is a first sample of horizontal deflection in a sequence, x_2 is a second sample of horizontal deflection in a sequence, y_1 is a first sample of vertical deflection in a sequence, y_2 is a second sample of vertical deflection in a sequence, etc.), or change in vector direction, is determined by comparing the digitized horizontal and vertical deflection signals in a digital sample of a vector to the digitized horizontal and vertical deflection signals in the previous digital sample of a vector. The start and end points of a vector are determined from changes in vector direction, from changes in color, from changes in intensity, from changes in draw rate (distance between two digital samples of a vector divided by the sample period), or from changes in other digitized signals in a digital sample of a vector. Also, the present invention determines whether a plurality of small samples combine to result in one large vector (i.e., many small vectors can appear to be one vector). It is noted that vectors can be detected by determining end points from stroke-like video, by recognition of vectors in a raster image, or by parsing vector draw commands in a graphics array or display list in a manner known to those skilled in the art.

The present invention declares a change in vector direction when a measured change in direction exceeds a predetermined static or dynamic threshold value (which is affected by noise or the processor characteristics) in a manner that that will be apparent to those skilled in the art. The length of a vector is determined from the distance between the start and end points in accordance with the following equation: $\text{sqrt}((x_2 - x_1)^2 + (y_2 - y_1)^2)$. The intensity of a vector is determined from the digitized intensity signal in a digital

sample of a vector and from the draw rate for a digital sample of a vector. A slower draw rate corresponds to a higher intensity for the vector. A vector primitive is then formed by combining the start and end points, length, direction, color, and intensity of a vector. Vector primitives are added to a vector array (Graphics Array 260) for each image or video frame of vectors to be rendered on, for example, a color flat panel display 630. The beginning and end of a video frame of vectors is determined directly from stroke video signals or other synchronizing information, by detecting a vector primitive already in the vector (graphics) array 260, by a timer, or by other means. Changes in other vector measurements are declared in a similar manner and will become apparent to those skilled in the art.

Referring to FIG. 3, there is shown a Graphics Detection Processor 200 in accordance with the present invention. The Graphics Detection Processor 200 can be programmable in order to accept multiple forms of Rendered Graphics 220 as input. Rendered Graphics 220 can be imagery, video, graphics, graphics commands, or other appropriate input that are generated by a particular source such as stroke deflection processor, raster image processor, image scanner, still or motion camera, or a graphics application. The ability to program the Graphics Detection Processor 200 provides for optimization for input types, input content, performance, security, transmission, storage, image quality, specific display characteristics, and other purposes as will become apparent to those skilled in the art. Programming can be based on defaults or dynamically assigned values for optimization in a manner known to those skilled in the art.

Whereas conventional systems provide as output a bit map from Rendered Graphics 220, the Graphics Detection Processor 200 provides as output a Graphics Array 260 that represents the graphics information provided in Rendered Graphics 220. The generation of a Graphics Array 260 is one novel component of the present invention. A Graphics Array 260 is also referred to herein as a display list. Those skilled in the art can render graphics with a Graphics Array 260 in an optimal manner for a given display. Acquisition 230 module converts Rendered Graphics 220 to a form suitable for processing. For example, if Rendered Graphics 220 is in the form of analog stroke video, Acquisition 230 can convert Rendered Graphics 220 to digital samples which

can then be processed by a digital processor. Recognition 240 module identifies graphics objects within Rendered Graphics 220. For example, Recognition 240 can determine start and end points of vectors, position and type for graphics objects such as circles and spheres, or position, type, and font for characters in the digital samples provided by Acquisition 230. Array Generator 250 collects the information for each graphics object identified by Recognition 240 module and creates a Graphics Array 260 with information for each graphics object or primitive. For example, Graphics Array 260 can be a display list of vectors where each entry in the display list includes the following information: start point (x, y), end point (x, y), color (red level, green level, blue level), intensity (voltage or relative brightness level), draw rate (inches/second). In Graphics Array 260 the order of appearance in an array can be used to determine graphics object precedence or other processing functions.

Referring to FIG. 4, there is shown an Encode Processor 300 in accordance with the present invention. The Encode Processor 300 receives Graphics Array 260 as input. The Encode Processor 300 provides as output an Optimized Graphics Array 330 that represents the graphics information in Graphics Array 260. The Encode Processor 300 can be optimized for input types, input content, performance, security, transmission, storage, image quality, specific display characteristics, and other purposes. As a non-limiting example, the Encode Processor 300 can perform vector quantization, data compression, data encryption, graphics object sorting, alpha value assignment (e.g., precedence or transparency), or other processing functions. As a further example, the Encode Processor 300 can filter each vector in a Graphics Array 260 to ensure that all vectors are of reasonable length, color, and intensity in order to optimize their appearance on the color flat panel display. As another example, Encode Processor 300 can encode Graphics Array 260 as an array of graphics commands or graphics routine calls such as those supported by OpenGL® computer software available from Silicon Graphics, Inc. As another example, Encode Processor 300 can encode a chain of vectors which form a circle as a single draw circle command, which is also the case for Recognition 240. The output from the Encode Processor 300 is the Optimized Graphics Array 330.

Referring to FIG. 5, there is shown a Graphics Rendering Processor 400 in accordance with the present invention. The Graphics Rendering Processor 400 receives as input a Graphics Array 260 or an Optimized Graphics Array 330 from the Encode Processor 300. The Graphics Rendering Processor 400 provides as output Display Data 530 which is a rendering (processing/formatting) of the graphics information contained within a Graphics Array 260 or an Optimized Graphics Array 330. Display Data 530 is appropriately formatted for a graphics display. Display Data 530 can contain any or all of the following types of information: bit map, texture map, raster graphics, vector graphics, holographics and other graphics formats. The Graphics Rendering Processor 400 can perform graphics processing functions including anti-aliasing in order to smooth lines, texturing to enhance the appearance of graphics objects, alpha blending to combine graphics objects and maintain correct color perception, haloing to increase contrast between graphics objects and the background scene, interpolation for scaling and smoothing graphics objects, fogging to provide the perception of depth, fills in order to color graphics objects, merging with other graphics information and sources, and other functions. The Graphics Rendering Processor 400 can be optimized for input types, input content, performance, security, transmission, storage, image quality, specific display characteristics, and other purposes.

The Graphics Rendering Processor 400 can also be optimized for a color flat panel display 600. The Graphics Rendering Processor 400 processes the Graphics Array 260 or Optimized Graphics Array 330 by carrying out a graphics render command for each object or primitive in the array. Priority for vector primitive rendering is determined by the position of a vector primitive in the array, where the first primitive (or other predetermined value) has the highest priority. Priority can also be assigned based on the type of graphics object recognized by Recognition 240. The Graphics Rendering Processor 400 operates in the raster domain and applies anti-aliasing, texturing, alpha blending, haloing, interpolation, fogging, shading, fills, other rendering techniques, and combinations thereof, as it renders individual primitives. The output from the Graphics Rendering Processor 400 is Display Data 530. Display Data 530 contains formatted visual information ready for transmission

to a display or other device (e.g., storage device). For example, Display Data 530 can be a bit map formatted for a color flat panel display.

Display Data 530 is transmitted over, for example, a pixel bus to a color flat panel Display 600 in a manner well known to those skilled in the art where it can be viewed by a person. Those skilled in the art will recognize, however, that any means of transmitting information from one location to another will operate in the present invention.

Referring to FIG. 6, there is shown a Display 600. A Display 600 receives as input Display Data 530. A Display 600 provides as output a Display Surface/Volume 630 which is a representation of Rendered Graphics 220. Display Surface/Volume 630 represents the desired presentation of Rendered Graphics 220. The representation of Display Surface/Volume 630 can be viewable information, transmitted information, stored information, or other appropriate form of information. A Display Surface/Volume 630 can be a cathode ray tube device, flat panel device, liquid crystal display device, projection device, holographic device, retinal projection device, storage device, printer, transmitter, or any other device or method for presenting, storing, transmitting, conveying, or making graphics information viewable, accessible, or usable.

Other forms of information can be re-generated using the method and system embodied in the present invention. For example, referring to FIG. 7 there is shown an alternate embodiment 800 of the present invention for this purpose. Information Detection Processor 820 is a more generalized form of the Graphics Detection Processor 200. Encode Processor 830 in FIG. 7 is a more generalized form of the Encode Processor 300 described earlier. Information Processor 840 in FIG. 7 is a more generalized form of the Graphics Rendering Processor 400. Other Information 810 in FIG. 7 is a more generalized form of the Display Formatter 1100 (FIG. 10). Merge Processor 850 in FIG. 7 is a more generalized form of the Merge Processor 500 described herein. Information Storage/Transmission 860 in FIG. 7 is a more generalized form of the Display 600 of FIG. 6.

Having described one embodiment of the present invention, attention is now turned to an alternative embodiment of the graphics rendering apparatus and method previously described. In particular, reference is made to FIG. 8

where there is shown an alternate embodiment 20 of the present invention represented by its major components. The block diagram of FIG. 8 represents an alternate embodiment of the invention shown in FIG. 7. The discussion of the Graphics Detection Processor 200 and of Encode Processor 300 is the same as that provided earlier with respect to FIG. 3 and 4, respectively, and, accordingly, will not be discussed again.

Referring to FIG. 9, there is shown a Graphics Processing 1000 in accordance with the alternate embodiment of the present invention. The Graphics Processor 420 accepts as input a Graphics Array 260 or an Optimized Graphics Array 330. The Graphics Processor 420 provides as output Frame Buffer 430 which is a rendering of the graphics information contained within a Graphics Array 260 or an Optimized Graphics Array 330. Frame Buffer 430 can be a memory device for display systems that scan out of frame buffer memory, or a stream of information provided directly to Merge Processor 500 or Display 600 for display systems that employ a flow-through method. Frame Buffer 430 is appropriately formatted for a particular type of a graphics display. Frame Buffer 430 can contain any or all of the following types of information: bit map, texture map, raster graphics, vector graphics, holographics, and other graphics formats. The Graphics Processor 420 can perform certain graphics processing functions including anti-aliasing, texturing, alpha blending, haloing, interpolation, fogging, fills, merging with other graphics information and sources, and other functions. The Graphics Rendering Processor 1000 can be optimized for input types, input content, performance, security, transmission, storage, image quality, specific display characteristics, and other purposes.

Referring to FIG. 10, there is shown a Display Formatter 1100 in accordance with the alternate embodiment 800 of the present invention. The Acquisition 135 accepts as input 110 imagery, video, graphics, or graphics commands. The Formatter 130 receives the information from Acquisition 135 and provides as output Frame Buffer 140. Frame Buffer 140 includes a rendering of the imagery, video, graphics, or graphics commands appropriately formatted for a graphics display.

Referring to FIG. 11, there is shown a Merge Processor 500 in accordance with the alternate embodiment 800 of the present invention. The Merge Processor 500 accepts as input rendered graphics in multiple frame

5 buffers 1 through n (510 and 540), combines the contents of the frame buffers in Combine 520, and provides as output Display Data 530. Display Data 530 is a combined rendering of graphics from any or all frame buffers appropriately formatted for a graphics display (display data 530). The Merge Processor 500 provides maximum flexibility in combining graphics information from many sources for a graphics display. The Merge Processor 500 can perform other graphics functions including anti-aliasing, texturing, alpha blending, haloing, interpolation, fogging, fills, merging with other graphics information and sources, and other functions. The Merge processor 500 is an optional component of the present invention in accordance with the alternate embodiment.

10 Referring again to FIG. 6, there is shown the Display 600 for use in the alternate embodiment. The discussion with respect to FIG. 6 above applies to this alternate embodiment, and accordingly, need not be discussed again. In the alternate embodiment, however, the output, Display Surface/Volume 630, is a representation of Rendered Graphics 220 combined with input 110. Display 630 represents the desired presentation of Rendered Graphics 220 combined with input 110.

20 *Example 1*

The following non-limiting example is provided to illustrate the operation of the alternate embodiment 20 as applied to stroke symbology to raster symbology conversion for use in a flat panel display. FIG. 8 shows functionally an implementation of the present invention to solve the problem of displaying high-quality, anti-aliased color stroke symbology along with high-quality color raster video on a color flat panel display.

30 Referring again to FIG. 3, the Graphics Detection Processor 200 detects the individual vectors provided by a legacy stroke (vector) video interface for each video frame. The Graphics Detection Processor 200 digitizes the horizontal and vertical deflection signals, color signals, intensity signals, symbology blanking signals and other signals from the stroke (vector) video using analog to digital converters to form digital samples of vectors. The direction of a vector is determined by comparing the digitized horizontal and vertical deflection signals in a digital sample of a vector to the digitized

horizontal and vertical deflection signals in the previous digital sample of a vector. The start and end points of a vector are determined from changes in vector direction, from changes in color, from changes in intensity, from changes in draw rate, or from changes in other digitized signals in a digital sample of a vector. A change in vector direction is declared when a measured change in direction exceeds a static or dynamic threshold value. Changes in other vector measurements are declared in a similar manner. The length of a vector is determined from the distance between the start and end points. The intensity of a vector is determined from the digitized intensity signal in a digital sample of a vector and from the draw rate for a digital sample of a vector. A slower draw rate corresponds to a higher intensity for the vector. A vector primitive is formed by combining the start and end points, length, direction, color, draw rate, and intensity of a vector. Vector primitives are added to a vector (graphics) array 260 for each video frame of vectors and prioritized to be rendered on a color flat panel display 630. The beginning and end of a video frame of vectors is determined directly from stroke video signals or other synchronizing information, by detecting a vector primitive already in the vector (graphics) array 260, by a timer, or by other means.

Referring again to FIG. 4, the Encode Processor 300 filters each vector in the vector array 260 to ensure that all vectors are of reasonable length, color, and intensity in order to optimize their appearance on the color flat panel display. For example, the measured color for a vector primitive can be encoded as a color optimized for presentation on the color flat panel display and for other considerations like night vision equipment compatibility. Also, the intensity of a vector primitive can be encoded to take into account the effects of draw rate or the desire to assign discrete intensity levels based on precedence or other criteria. In addition, the Encode Processor 300 can format information associated with each vector primitive in a manner optimized for the creation of a graphics rendering command. The output from the Encode Processor 300 is the optimized vector (graphics) array 330.

Referring again to FIG. 9, the Graphics Rendering Processor 1000 is optimized for a color flat panel display 600. The Graphics Rendering Processor 1000 processes the optimized vector (graphics) array 330 by issuing a vector (graphic) render command to a Graphics Processor 400 for each vector

primitive. Priority for vector primitive rendering is determined by the position of a vector primitive in the optimized vector (graphics) array 330, where the first vector primitive in the optimized vector (graphics) array 330 has the highest priority. The graphics processor operates in the raster domain and applies anti-aliasing, texturing, alpha blending, haloing, interpolation, fogging, and fills as it renders individual vector primitives. The output from the Graphics Rendering Processor 400 is rendered vectors (Frame Buffer) 430.

Referring again to FIG. 10, the Display Formatter 1100 accepts input 110 analog raster video requiring minimal processing by Acquisition 135 for the purpose of rendering on the color flat panel display 630. Acquisition 135 can perform functions like analog to digital conversion, intensity and display size scaling, and other functions well understood by those skilled in the art. The output from the Formatter 130 is formatted raster video (Frame Buffer) 140.

Referring again to FIG. 11, the Merge Processor 500 combines the rendered vectors (Frame Buffer) 530 from the Graphics Rendering Processor 400 with the formatted raster video (Frame Buffer) 140 from the Display Formatter 1100 in order to provide a more comprehensive display format to view. The output from Merge Processor 500 is Display Data 530.

Example 2

The following non-limiting example is provided to illustrate the operation of the alternate embodiment 20 as applied to stroke to raster conversion for use in a flat panel display. FIG. 12 shows functionally an implementation of the alternate embodiment 20 of the present invention to solve the problem of displaying high-quality, anti-aliased color stroke symbology along with high-quality color raster video on a color flat panel display.

The present invention detects the individual vectors drawn by the stroke generator, for example, a Multipurpose Display Indicator (MDI) or display computer, with the Stroke Vector Detector. This function effectively creates the original symbology display list used by the stroke generator.

Stroke symbology vectors are then rendered and anti-aliased by Vector Rendering. This function takes advantage of off-the-shelf graphics rendering components using commercially available, state-of-the-art rendering algorithms. The combination of display list recreation and vector rendering by Stroke Vector

Detector and Stroke Vector Rendering, respectively, is a novel component of the present invention. Indeed, the present invention is capable of displaying stroke symbology with far greater quality and accuracy than any currently-available stroke generator is capable of providing it. Raster video is digitized and scaled by Raster Digitize/Scaling. This function is equivalent to that found in most flat panel displays and displays processing systems with analog video. Rendered stroke symbology and digitized raster video are merged in Merge. Finally, the merged video is displayed on the Display.

The stroke vector detection function performed by the Stroke Module are shown in FIG. 13. Deflection and bright up signals (Stroke Video) are switched by Stroke Switching. Stroke Switching also provides a repeater function for Stroke Video. The selected Stroke Video input is digitized by the Analog To Digital Converter at a, for example, 48 MHz sample rate. This sample rate corresponds to a minimum of 4 samples per display increment rendered at the fastest writing rate, or about 7 samples per pixel on the display. The digitization of stroke signals provides an effective symbology resolution of at least, for example, 4800 by 4800 pixels with a 12 bit A/D, and possibly, for example, 9500 by 9500 pixels with a 13 bit A/D. Once digitized, Vector Recognition determines the start and end points, color, and intensity of each vector within Stroke Video symbology and places this vector information into Display List. Symbology quality is maximized by ensuring that the same point is used for the end point of one vector and the start point of the next vector in a vector sequence, like that in an arc. Variability in symbology writing rates are automatically accounted for with this method. In addition, variability in draw rate can be used to modify symbology intensity is required.

The Stroke Vector Detector can be contained in a single programmable logic device, array, or application specific integrated circuit (ASIC).

As shown in FIG. 14, Stroke symbology vectors are then rendered and anti-aliased from the Display List by Vector Rendering under control from the Display Control module described herein (not shown). The Frame Buffer is filled with high-quality, anti-aliased symbology ready for display. Stroke symbols drawn first in a frame are typically the highest priority symbols and could be rendered accordingly through alpha blending and other graphics functions.

The Stroke Module enables the present invention to render stroke symbology with unprecedented accuracy and quality. This is possible because Vector (Graphics) Rendering enables the present invention to process stroke symbology with anti-aliasing using an array of 16 by 16 color sub-pixels per display pixel. Visually, this translates to the ability to render each stroke symbology pixel within, for example, a 9500 by 9500 pixel virtual display registered to a, for example, 600 by 600 pixel actual display. These levels of pixel and sub-pixel processing resolution are necessary in order to display the highest possible quality symbology.

The functions performed by the Raster Module are shown in FIG. 15. Raster video signals (Raster Video) are switched by Raster Switching. Raster Switching also provides a repeater function for Raster Video. The selected Raster Video input is digitized by the Analog To Digital Converter. A Sync Detect and phase locked loop (PLL) apparatus (not shown) performs synchronization detection on the incoming Raster Video and generates Raster Module clocks.

A commercially-available, off-the-shelf integrated circuit and low cost field memory can be used to perform vertical/temporal de-interlacing, image scaling, and gamma correction. The present invention can be also adapted to provide a growth provision for zoom in a manner that will become apparent to those skilled in the art.

The Display Controller (not shown) performs processing functions and controls all modules of the present invention with plenty of available processing and throughput overhead. One important task performed by this module is the execution of built in tests. Another important task performed by this module is accessing the Display List created by the Stroke Module in order to form graphics commands for Vector (Graphics) Rendering on the Stroke Module.

A Liquid Crystal Display (LCD) assembly suitable for use in the present invention can be, for example, a high-resolution, state-of-the-art design with a 600 by 600 color pixel Active Matrix Liquid Crystal Display (AMLCD) having 120 color groups per inch. Other raster-type displays can be used as will become apparent to those skilled in the art. Also, the present invention has multiple applications, including raster scanned or calligraphically-generated cathode ray

tube (CRT) displays, X-Y plotters, numerically-controlled machines, robotics, etc.

To create an analog video signal from digital raster data, a video display generator retrieves the digital data from memory to create a digital signal. The display generator then creates an analog signal from the digital signal with a digital-to-analog (D/A) converter. The analog signal is then amplified and displayed as a video image.

During the video image creation process, many factors adversely affect the analog video signal output from the display generator. For example, signal interpolation within the D/A converter introduces distortions to the analog signal. These distortions are then magnified during amplification of the signal. In addition, the analog amplifier, which functions differently at different signal frequencies, creates further distortion in the analog output. Because of the non-linearity introduced by the elements of the video display generator, the output analog signal is not a completely accurate representation of the digital data from which it was created. Although perfectly accurate video images cannot be expected, it is often necessary to ensure that the displayed image is as accurate a representation of the digital data as possible. Therefore, an all digital interface is preferred between the frame buffer and the display surface/volume, eliminating the conversion errors and distortions discussed herein.

The present invention is subject to many variations without departing from the spirit and scope of the present invention. For example, a digital map analog or digital interface can be added as an additional raster input in order to enable the system to overlay symbology onto digital map video. The resulting combination of the symbology with digital map is then provided through an analog or digital repeater.

Additionally, a digital map rendering module can be embedded in order to enable the system to overlay symbology onto an internally rendered digital map. The resulting combination of symbology with digital map is then provided through an analog or digital repeater. An interface to a digital map mass memory unit can also be added.

The modular design of the present invention along with available processing and throughput overhead provide for maximum flexibility for future

growth. For example, a modification to the Status/Control interface or the addition of a new data interface, along with Display Controller software modifications, would enable the system to become a smart display. A smart display enables raster symbology to be rendered in manner optimized for the display. Other functions and features such as zoom, alpha blending, and rendering stroke symbology with halos can also be accommodated in a manner that will become apparent to those skilled in the art.

Other variations and modifications of the present invention will be apparent to those of skill in the art, and it is the intent of the appended claims that such variations and modifications be covered. For example, text from books or other printed media can be converted to other fonts or even other languages. Graphics can be re-rendered and their quality improved upon. Two-dimensional pictures of three-dimensional objects can be converted to three-dimensional virtual reality images. A system for converting books to electronic media can completely re-render all information in the book with the target media in mind. A single video source can be used to drive multiple display types, even simultaneously, where rendering is performed to optimize the video presentation for each display type.

The particular values and configurations discussed above can be varied and are cited merely to illustrate a particular embodiment of the present invention and are not intended to limit the scope of the invention. It is contemplated that the use of the present invention can involve components having different characteristics as long as the principle is followed. It is intended that the scope of the present invention be defined by the claims appended hereto.

CLAIMS

1. A method of detecting graphic objects that have been created for a first display, the graphic objects representing an image, and re-rendering the graphic objects in a form adapted for a different display, the method comprising
5 the steps of:
 - receiving formatted graphics from a source, the formatted graphics having been formatted for the first display;
 - detecting a plurality of graphic objects within the formatted graphics;
 - 10 creating a graphics array from the plurality of graphics objects that represents the image based on the formatted graphics; and
 - re-rendering the plurality of graphics objects in a manner formatted for the different display.
- 15 2. The method of Claim 1, wherein the graphics objects are graphics primitives.
3. The method of Claim 2, wherein the graphics primitives are vector primitives.
- 20 4. The method of Claim 1, wherein the graphics objects are vector primitives.
5. The method of Claim 1, wherein the graphics objects are combinations
25 of a plurality of graphics primitives.
6. The method of Claim 5, wherein each of the plurality of graphics primitives are a plurality of vector primitives.
- 30 7. The method of Claim 1, further comprising the step of displaying the re-formatted graphics objects on the different display.
8. The method of Claim 1, further comprising the step of storing the re-formatted graphics objects for subsequent display on the different display.

9. The method of Claim 1, further comprising the step of transmitting the re-formatted graphics objects for use by the different display.

5 10. The method of Claim 1, wherein the graphics array comprises characteristics of a portion of the image.

10 11. The method of Claim 10, wherein the characteristics are selected from the group consisting of color, intensity, size, location, transparency, texture, shape, direction, precedence of a portion of the image and combinations thereof.

12. The method of Claim 1, wherein the graphics array comprises a plurality of vector primitives of a portion of the image that are compatible with a graphics processor particular to the different display.

15

13. The method of Claim 1, wherein the graphics array is a display list.

14. The method of Claim 1, further comprising the step of applying anti-aliasing techniques to the graphics objects within the graphics array.

20

15. The method of Claim 1, further comprising the step of applying alpha blending techniques to the graphics objects within the graphics array.

25 16. The method of Claim 1, further comprising the step of applying haloing techniques to the graphics objects within the graphics array.

17. The method of Claim 1, further comprising the step of applying interpolation techniques to the graphics objects within the graphics array.

30 18. The method of Claim 1, further comprising the step of applying fogging techniques to the graphics objects within the graphics array.

19. The method of Claim 1, further comprising the step of merging the re-rendered plurality of graphics objects with graphics information from sources other than the first display or different display.

5 20. The method of Claim 1, wherein the graphics objects can be characterized by a plurality of vectors, and further comprising the step of detecting each individual vector within the plurality of vectors.

10 21. The method of Claim 20, further comprising the step of detecting a start point and an end point for each individual vector.

22. The method of Claim 21, wherein the start point and the end point are detected by sampling the individual vector.

15 23. The method of Claim 20, wherein the step of detecting each individual vector within the plurality of vectors further comprises the step of detecting a change in direction of the graphics objects.

20 24. The method of Claim 20, wherein the step of detecting each individual vector within the plurality of vectors further comprises the step of detecting a change in color of the graphics objects.

25 25. The method of Claim 20, wherein the step of detecting each individual vector within the plurality of vectors further comprises the step of detecting a change in intensity of the graphics objects.

26. The method of Claim 20, wherein the step of detecting each individual vector within the plurality of vectors further comprises the step of detecting a change in draw rate of the graphics objects.

30 27. The method of Claim 20, further comprising the step of detecting a change in direction of each individual vector.

28. The method of Claim 27, wherein the change in direction is detected by comparing a measured change in direction and a predetermined threshold value of direction.

5 29. The method of Claim 21, further comprising the step of creating a vector primitive for each individual vector by combining the start point and the end point and a plurality of characteristics of each individual vector.

10 30. The method of Claim 29, wherein the plurality of characteristics of each individual vector comprise length, direction, color, draw rate, and intensity of each individual vector.

15 31. The method of Claim 29, further comprising the step of adding the vector primitive to the graphics array in a predetermined order.

32. The method of Claim 1, further comprising the step of optimizing the graphics array for the different display.

20 33. The method of Claim 1, further comprising the step of prioritizing the plurality of graphics objects within the graphics array.

25 34. An apparatus for detecting graphic objects that have been created for a first display, the graphic objects representing an image, and re-rendering the graphic objects in a form adapted for a different display, the apparatus comprising:

receiving means for receiving formatted graphics from a source, the formatted graphics having formatted for the first display;

detecting means, in communication with said receiving means, for detecting a plurality of graphic objects within the formatted graphics;

30 generation means, in communication with said detecting means, for creating a graphics array from the plurality of graphics objects that represents the image based on the formatted graphics; and

re-rendering means, in communication with said generation means, for re-rendering the plurality of graphics objects formatted for the different display.

35. The method of Claim 34, wherein the graphics objects are graphics primitives.

5 36. The method of Claim 35, wherein the graphics primitives are vector primitives.

37. The method of Claim 34, wherein the graphics objects are vector primitives.

38. The method of Claim 34, wherein the graphics objects are combinations of a plurality of graphics primitives.

10

35. The apparatus of Claim 1, wherein the detecting means comprises means to digitize the graphics objects.