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

(11) EP 0 984 392 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
**23.06.2004 Bulletin 2004/26**

(21) Application number: **98921774.0**

(22) Date of filing: **22.05.1998**

(51) Int Cl.[7]: **G06T 1/00**, G06T 11/40, G06T 15/50

(86) International application number:
**PCT/JP1998/002262**

(87) International publication number:
**WO 1998/053425 (26.11.1998 Gazette 1998/47)**

(54) **IMAGE PROCESSOR AND IMAGE PROCESSING METHOD**

BILDPROZESSOR UND BILDVERARBEITUNGSVERFAHREN

PROCESSEUR D'IMAGE ET PROCEDE CORRESPONDANT

(84) Designated Contracting States:
**DE ES FR GB IT**

(30) Priority: **22.05.1997 JP 13183197**

(43) Date of publication of application:
**08.03.2000 Bulletin 2000/10**

(73) Proprietor: **SEGA ENTERPRISES, LTD.**
**Tokyo 144-0043 (JP)**

(72) Inventors:
- **YASUI, Tetsuhiro**
  **Ohta-ku Tokyo 144-0043 (JP)**
- **MORIOKA, Seisuke**
  **Ohta-ku Tokyo 144-0043 (JP)**
- **OKUBO, Jun**
  **Ohta-ku Tokyo 144-0043 (JP)**

(74) Representative: **Brown, Kenneth Richard et al**
**R.G.C. Jenkins & Co.**
**26 Caxton Street**
**London SW1H 0RJ (GB)**

(56) References cited:
**JP-A- 3 071 378**
**JP-A- 3 201 081**
**JP-A- 4 170 686**
**US-A- 4 885 703**



EP 0 984 392 B1

Printed by Jouve, 75001 PARIS (FR)

## Description

Technical Field:

**[0001]** The present invention relates to an image processor and an image processing method for computer graphics.

Background Art:

**[0002]** In the field of computer graphics, to realize a higher capability of image processing, there has been known a method of processing by which a screen is divided into tiny rectangular regions (hereinafter referred to as "fragments") and processing is done fragment by fragment.

**[0003]** This conventional method will now be described with reference to Figs.14 to 17, wherein, at first, (1): each polygon is divided by frames of fragments and memorized with the fragments containing the polygons known.

**[0004]** In Fig.14, there are fifteen fragments arranged in a matrix of 3x5, on which polygons A, B and C are defined. Concerning the polygon A, it corresponds to eight fragments, as shown in Fig.15. These fragments are stored in a buffer memory shown in Fig.16 according to their coordinates $\alpha$, $\beta$, $\gamma$, etc. $\alpha 1$ and $\alpha 2$ in Fig.15 are stored in the memory area $\alpha$ of the buffer memory, $\beta 1$ and $\beta 2$ in Fig.15 are stored in the memory area $\beta$ of the buffer memory, and $\gamma$ 1 and $\gamma$ 2 in Fig.15 are stored in the memory area $\gamma$ of the buffer memory. The same is applied to the remaining coordinates.

**[0005]** (2): Next, of all the fragments, fragments covered by a polygon currently concerned are computed and painted. For example, as shown in Fig.17, a fragment $\delta 1$ is first painted toward the polygon A (see Fig. 17(b)), painted toward the polygon B (see Fig.17(c)), and finally painted toward the polygon C (see Fig.17(d)). Such operation is repeated until the final polygon is processed or the fragment is totally covered. And the forgoing processing is performed for all the fragments.

**[0006]** The conventional data processing of image elements has been performed on the foregoing procedures (1) and (2).

**[0007]** However, such method requires all the polygons to be divided into objective fragments, resulting in that a huge amount of calculation is necessary and a long processing time is also required.

**[0008]** Additionally, there arise various drawbacks. For example, a storage of a greater capacity needs to be arranged to memorize information to be produced about polygons which have been divided. In order to gain faster processing, the storage should be faster in operation, raising the cost for the storage.

**[0009]** In cases where semitransparent polygons possess texture data, these can be processed by the conventional method.

**[0010]** The present invention is intended to resolve the foregoing problems, and aims to enable processing of data of image elements to be performed efficiently and rapidly in computer graphic systems and to obtain higher-quality images at a lower cost than in conventional systems. One method to realize such an aim is based on dividing a screen into tiny regions (fragments) for processing, and efficiently retrieving polygons included in regions concerned. Briefly, a system and a method according to the invention can provide images of higher quality at a lower cost than in the conventional systems.

Summary of the Invention:

**[0011]** In one aspect, the invention provides an image processor in which a screen is divided into regions of a predetermined size and processing on polygon data is performed region by region, comprising first rearranging means for providing constituting elements of a plurality of polygons, falling into individual regions on the screen in a direction vertical to a scanning line, with vertical links mutually linking constituting elements of a polygon, and for rearranging links to constituting elements falling into the same region, thus linking these elements; second rearranging means for providing constituting elements of a plurality of polygons, falling into individual regions on the screen in the direction of the scanning line, with horizontal links mutually linking constituting elements of a polygon, and for rearranging links to constituting elements falling into the same region, thus linking these elements; and an image processing means for retrieving and processing constituting elements of diverse polygons falling in a given region, the sequence of constituting elements being given by rearranged links.

**[0012]** In the described embodiment, the first rearranging means is operable to perform the rearrangement based on minimum or maximum values of the constituting elements with reference to the vertical direction; and the second rearranging means is operable to perform the rearrangement based on minimum or maximum values of the constituting elements with reference to the horizontal direction, in vertical coordinates of a region to be processed by the first rearranging means.

**[0013]** It is thus clear that the rearrangement can be performed either by minima or maxima, only the directions in the order of processing being changed ("left to right sides" to "right to left sides" or "upper to lower sides" to "lower to upper sides"), as necessary. Also, similar processing can be used whether the processing order is vertical to horizontal sides or horizontal to vertical sides.

**[0014]** In the embodiment, the first and second rearranging means determine whether or not, after completion of the processing of a polygon in the image processing means for a said region, the constituting elements for said polygon to be used by the image processing means are included in a region to be image-processed

at the next processing, and perform link update processing which removes from the link the constituting elements which are determined to not be included.

**[0015]** In the embodiment, the image processing means, as to image processing for each divided region, not only sections polygons to be processed into opaque polygons, polygons with transparent pixels, and semitransparent polygons but also processes in the order of the opaque polygons, polygons with transparent pixels, and semitransparent polygons.

**[0016]** In another aspect, the invention provides an image processing method by which a screen is divided into regions of a predetermined size, and processing on polygon data is performed region by region,the method comprising the steps of a first rearranging step of providing constituting elements of a plurality of polygons, falling into individual regions on the screen in a direction vertical to a scanning line, with vertical links mutually linking constituting elements of a polygon, and rearranging links to constituting elements falling into the same region, thus linking these elements; a second rearranging step of providing constituting elements of a plurality of polygons, falling into individual regions on the screen in the direction of the scanning line, with horizontal links mutually linking constituting elements of a polygon, and rearranging links to constituting elements falling into the same region, thus linking these elements; and an image processing step of retrieving and processing constituting elements of diverse polygons falling in a given region, the sequence of constituting elements being given by rearranged links.

**[0017]** According to the described embodiment, in the first rearranging step, the rearrangement is performed based on minimum or maximum values of the constituting elements with reference to the vertical direction, and in the second rearranging step, the rearrangement is performed based on minimum or maximum values of the constituting elements with reference to the horizontal direction, in vertical coordinates of a region to be processed in the first rearranging step. According to the embodiment, in the first and second rearranging steps, link update processing is performed to invalidate an unnecessary region among the regions corresponding to the constituting elements.

**[0018]** According to the embodiment, in the image processing step, the image processing for each divided region is performed with polygons to be processed sectioned into opaque polygons, polygons with transparent pixels, and semitransparent polygons, and performed in the order of the opaque polygons, polygons with transparent pixels, and semitransparent polygons.

Brief Description of the Drawings:

**[0019]**

Fig.1 is a block diagram functionally outlining an image processor according to an embodiment of the present invention.

Fig.2 is a functional block diagram of a geometry processor of the image processor according to the embodiment of the present invention.

Fig.3 is a functional block diagram of a fill processor of the image processor according to the embodiment of the present invention.

Fig.4 is a functional block diagram of a texture processor of the image processor according to the embodiment of the present invention.

Fig.5 is a functional block diagram of a shading processor of the image processor according to the embodiment of the present invention.

Fig.6 is a flowchart showing the whole processing of the embodiment of the present invention.

Fig.7 is an illustration of processing as to a Y index buffer and a Y sort buffer in the embodiment of the present invention.

Fig.8 is an illustration of processing as to a Y index buffer and a Y sort buffer in the embodiment of the present invention.

Fig.9 is an illustration of processing as to a Y index buffer and a Y sort buffer in the embodiment of the present invention.

Fig.10 is an illustration of processing as to a Y index buffer and a Y sort buffer in the embodiment of the present invention.

Fig.11 exemplifies a fragment that first become valid for each polygon in the embodiment of the present invention.

Fig.12 is an illustration for link update processing in the embodiment of the present invention.

Fig.13 is an illustration for link update processing in the embodiment of the present invention.

Fig.14 is an illustration showing the relationship between fragments and polygons A, B and C, which explains the conventional processing.

Fig.15 is an illustration showing fragments that correspond to the polygon A, which explains the conventional processing.

Fig.16 is an illustration showing contents of a buffer memory, which is employed to explain the conventional processing.

Fig.17 is an illustration for painting processing according to the conventional processing.

**[0020]** An apparatus and method of an embodiment according to the present invention will now be detailed.

**[0021]** Fig. 1 is an outlined functional block diagram of an image processor according to the embodiment of the present invention. In the figure, a reference 1 is a CPU (central processing unit), which is used to operate for objects in a virtual space, to obtain operating information, and to perform a variety of controls. A reference 2 shows a geometry processor that performs high speed geometric conversion (i.e. calculation in vector quantities), such as coordinate conversion of polygons, clipping, perspective conversion in the three-dimensional

computer graphics and/or intensity calculation. A reference 2a is a polygon/material/light buffer memory (polygon/material/light buffer RAM), which is used to memorize valid polygon data, material data, and light data for one frame by the geometry processor 2 under operation. The polygons are polyhedrons constituting solids in the virtual space. The contents of data stored into the buffer memory 2a are as follows:

link information, coordinate information, and other attribute information about polygons, which include LINK X, LINK Y, X, Y, iz, Tx, Ty, Nx, Ny, Sign Nz, Alpha, Light ID, and Material ID;
information of materials, which includes Depth enable, Depth function, Depth density, Texture enable, Fog enable, translucency enable, texture type, texture function, offset x,y, size x,y, repeat x,y, mirror x,y, color id, Shinines, Material specular, Material emission, Polygon color, Texture mode, and blend mode; and
information of light, which includes Light position, Light Direction, Light Type, Attenuation Cutoff, Spotexp, Light Color, and Light Ambient.

**[0022]** A reference 3 is a fill processor that is in charge of hidden-surface elimination processing. The fill processor 3 paints polygons existing within a region in order to acquire pixel by pixel each piece of information on a polygon that is placed in the foreground position.

**[0023]** A reference 4 is a texture processor. The texture processor 4 maps textures on each pixel in a region. Texture mapping is processing that, for producing images, maps textures defined differently from the shape of an object onto the surface of the object of which shape is defined. A reference 4a is a texture memory (texture RAM), in which texture maps used in the processing of the texture processor 4 are preserved.

**[0024]** A reference 5 shows a shading processor. The shading is a technique that expresses shades or the like on objects composed of polygons in consideration of normal vectors of polygons, locations and colors of a light source, locations of a view point, directions of a line of sight, and others. The shading processor 5 calculates intensities of each pixel within a region. A reference 5a is a frame buffer into which image data for one screen are stored. Data are read in sequence from the frame buffer 5a, and then converted from digital data to analog signals, before supplied to such displays as a CRT, liquid crystal display, plasma display device, which are not shown in the figure.

**[0025]** A reference 6 represents a program work/polygon buffer memory (program work/polygon buffer RAM) memorizing commands (such as database of polygons and display lists) to a graphic processor as well as programs to the CPU 1. The buffer memory 6 also serves as a work memory for the CPU 1.

**[0026]** The fill processor 3, texture processor 4 and shading processor 5 are arranged for processing referred to as rendering. In each region, the rendering starts sequentially from its upper left position of the screen. Practically, the geometry enables objects to be located in the virtual space coordinates, and perspective-converts them onto a screen. By the rendering, pictures are produced by processing data defined on the screen coordinate. The rendering is repeated by the number of regions.

**[0027]** Referring to functional block diagrams shown in Figs.2 to 5, the image processor according to the embodiment of the present invention will now be detailed.

**[0028]** Fig. 2 is a functional block diagram of the geometry processor 2. In the figure, a reference 21 is a data dispatcher that not only reads commands from the buffer memory 6 but also analyzes them, controls a vector engine 22 and a clipping engine 24 based on the analyzed results, and provides a sort engine 27 with processed data.

**[0029]** A reference 22 shows a vector engine performing vector calculation. Vectors handled by the engine are stored in a vector register 23.

**[0030]** A reference 23 is a vector register, which preserves vector data which the vector engine 22 calculates.

**[0031]** A reference 24 is a clipping engine for clipping.

**[0032]** A reference 25 indicates a Y-sort index (Y-sort INDEX) preserving Y-indices used in Y-sorting executed by a sort engine 27.

**[0033]** A reference 26 expresses an X-sort index (X-sort INDEX) preserving X-indices used in X-sorting executed by a sort engine 27.

**[0034]** A reference 27 shows a sort engine that carries out X-sorting and Y-sorting, retrieving from the buffer 6 polygons falling into a fragment concerned. The retrieved polygons are not merely stored into the buffer memory 2a but also sent to the fill processor 3 for rendering. The sort engine 27 is also in charge of control of both a polygon TAG 28 and a polygon cache 34.

**[0035]** A reference 28 is a polygon TAG for preserving TAG of the polygon cache 34.

**[0036]** Fig.3 shows a functional block diagram of the fill processor 3. In the figure, a reference 31 is a cache controller controlling later-described material caches 42, 45, 51b, 52a and 53a and a light cache 51a.

**[0037]** A reference 32 is a material TAG, which preserves TAGs of later-described material caches 42, 45, 51b, 52a and 53a and a light cache 51a.

**[0038]** A reference 33 is a light TAG, which is a buffer preserving TAG of a light cache 51a described later.

**[0039]** A reference 34 represents a polygon cache, serving as a cache memory of polygon data.

**[0040]** A reference 35 shows an initial parameter calculator used for obtaining an initial value of DDA.

**[0041]** A reference 36 shows a Z-comparator array, which is responsible for the Z-comparison between polygons for hidden-surface elimination loading as well as processing polygon IDs and an interior division ratio of t0, t1 and t2. The Z-comparator array 36 is composed

of 64 (8x8) Z-comparators. These comparators operate in parallel, which enables 64 pixels to be processed concurrently. One Z-comparator preserves data of a polygon. For example, these data include polygon ID, iz, t0, t1, t2, window, stencil, and shadow.

**[0042]** A reference 37 is a vertex parameter buffer, which preserves parameters at the vertexes of polygons. Correspondingly to the Z-comparator array 36, the buffer has a capability of memorizing data of 64 polygons.

**[0043]** A reference 38 is an interpolator that calculates parameters of pixels by interpolation on the basis of the results t0, t1, t2 and iz calculated by the Z-buffer array 36 and the contents of the vertex parameter buffer 37.

**[0044]** Fig.4 is a functional block diagram of the texture processor 4. In the figure, a reference 41 is a density calculator calculating a blend ratio necessary for fog and depth queuing.

**[0045]** A reference 42 shows a material cache where data regarding depth information are stored.

**[0046]** A reference 43 indicates a window register to preserve pieces of information concerning windows. Such information is expressed in the form of kz, cz, fog function, fog density, and fog end z, for example.

**[0047]** A reference 44 is an address generator calculating addresses on a texture map using texture coordinates Tx, Ty and LOD.

**[0048]** A reference 45 shows a material cache in which data concerning materials are preserved.

**[0049]** A reference 46 is a TLMMI calculator (TLMMI: Tri Linear MIP Map Interpolation) that performs "Tri Linear MIP Map interpolation" adopted as three-dimensional interpolation. The "MIP Map" is a technique eliminating anti aliasing, that is, jags of textures in mapping textures. This technique relies upon the following principle. In principle, the color (intensity) of the surface of an object projected onto one pixel should be a mean value over the colors of corresponding mapping regions thereto. If not so, jags become apparent, severely deteriorating the quality of textures. Processing to obtain each mean value leads to enormous quantities of calculation load, with the result that processing takes a longer time or a fast processor is required. The MIP Map is a solution to this situation. According to MIP Map technique, to simplify collection of colors (intensities) of mapping regions in association with one pixel, a plurality of mapping data of which width is a multiple of two are prepared in advance. The sizes of all mapping regions in association with one pixel become values that exist between any two data produced by multiplying them by a multiple of two. By comparing these two data to each other, the colors of the corresponding mapping regions are obtained. For instance, if there are a one-time screen A and a 1/2-time screen B, each pixel of both the screens A and B corresponding to each pixel of a 1/1.5-time screen C is obtained. In this example, the color of each pixel of the screen C equals a medium color between the pixels of both the screens A and B.

**[0050]** A reference 47 is a color converter converting colors in 4-bit texel processing.

**[0051]** A reference 48 shows a color pallet in which color information is preserved in 4-bit texel processing. The color pallet 48 is used to store colors for graphic drawing. On the basis of the contents of the color pallet 48, a color available to each pixel is determined.

**[0052]** Fig.5 is a functional block diagram of the shading processor 5. In this figure, a reference 51 is an intensity processor that calculates intensities of polygons on which texture mapping has been completed.

**[0053]** A reference 51a is a light cache to memorize information on light.

**[0054]** A reference 51b shows a material cache to memorize information about materials. This information includes Shinies, Material specular, and material emission.

**[0055]** A reference 51c is a window register to preserve information about windows. Such information is formed from Screen center, Focus, Scene ambient, and others.

**[0056]** A reference 52 indicates a modulate processor that performs associating polygon colors with texture colors, intensity modulation, and fog processing.

**[0057]** A reference 52a is a material cache in which information on materials is stored. Such information includes Polygon color and Texture mode.

**[0058]** A reference 52b is a window register to preserve information on windows. This information includes Fog colors.

**[0059]** A reference 53 represents a blend processor that performs blending with data stored in a color buffer 54 and writes the result into the color buffer 54. The blend processor 53 blends a current pixel color and a pixel color of a frame buffer on the basis of a blend rate register, and then writes the blended result into a frame buffer of a bank represented by a write bank register.

**[0060]** A reference 53a is a material cache in which information regarding materials is stored. This information includes "blend mode."

**[0061]** A reference 54 shows a color buffer having the same size "8x8" as the fragments, which is made up of a double-bank structure.

**[0062]** A reference 55 is a plot processor that writes data in the color buffer 54 into the frame buffer 5a.

**[0063]** A reference 56 is a bitmap processor performing bitmap processing.

**[0064]** A reference 57 represents a display controller, which reads data in the frame buffer 5a, and provides them to a DAC (Digital to Analogue Converter) in order to show them on a not-shown display.

**[0065]** By the way, in the embodiment of the present invention, instead of dividing polygons fragment by fragment, polygons are rearranged in both the vertical direction (hereinafter referred to as the Y-direction) and the horizontal direction (hereinafter referred to as the X-direction) to a scanning line, resulting in that polygons included in each fragment can be retrieved. Using this

retrieved information makes it possible to process polygons every fragment concerned currently, enabling the processing without the foregoing division of the polygons. Accordingly, processing time is shortened and it is possible to realize the processing using an apparatus of which the memory device is smaller in capacity.

**[0066]** In rendering, the polygons are divided into types of opaque polygons, semitransparent polygons, and polygons with transparent pixels, and the semitransparent polygons are processed at the final step, which makes it possible to process the semitransparent polygons.

**[0067]** The entire flow of processing of the embodiment according to the present invention will be described in Fig.6.

**[0068]** Polygon data to be transmitted to the rendering section are rearranged by a minimum Y-coordinate value of the polygons, and written into the buffer (ST1).

**[0069]** Polygon data agreeing with the Y-coordinate value of a fragment concerned currently are rearranged by a minimum X-coordinate value of a concerned Y-region of polygons (ST2).

**[0070]** Rendering is performed toward all the opaque polygons whose coordinate agree with X- and Y-coordinates of a currently concerned fragment (ST3).

**[0071]** Rendering is performed toward all the transparent polygons whose coordinates agree with X- and Y-coordinates of a currently concerned fragment (ST4).

**[0072]** According to the data in the buffer, textures are painted, then shading is done, being converted into color data (ST5).

**[0073]** The steps ST4 and ST5 may be executed in the opposite order. Actually, it may be more practical to execute them in the order of steps ST5 to ST4.

**[0074]** Semitransparent polygons agreeing with X- and Y-coordinates of a fragment currently concerned are rendered in the order of Z-coordinate values (ST6).

**[0075]** Color data in the buffer are written into the frame buffer (ST7).

**[0076]** As shown in Fig.6, polygon data are first rearranged by Y, then the rearranged data are rearranged by minimum values of X in a region Y before proceeding to processing of each line. As a result, the position of a fragment of a polygon, which is to be processed at first, is found. Accordingly, it is first found which fragment includes the polygon. In addition, after this processing concerning the first fragment included, each polygon undergoes link update processing (later described), by which polygons unnecessary for the fragments to be processed subsequently are removed, so that only necessary polygons are included in pieces of information on polygons included in the next fragment. Accordingly tracing back all the links that have been made so far allows all the polygons falling into each fragment to be retrieved. Without dividing the polygons every fragment in order to produce newly processed polygons, information about polygons included in each fragment can be read and processed.

**[0077]** The link update processing is a technique that examines whether or not, after completing the processing of a polygon processing for a fragment concerned, the polygon becomes invalid in the fragments that will be subsequently concerned, and the examined polygon is removed from the link if it becomes invalid.

**[0078]** The processing unit for rendering will be explained below.

(1): The polygons are first divided into types of opaque polygons, polygons with transparent pixels, and semitransparent polygons.

(2): The opaque polygons are then subject to processing. Overwriting, independently of texture data, realizes this processing, producing valid pixels finally remaining. Only the Z-comparison is then performed at a faster speed.

(3): When having completed the above processes (1) and (2), all the data in the fragment buffer become valid. At this stage, texture data are read based on the data in the fragment buffer, then their color data are converted.

(4): Polygons with transparent pixels are then rendered. To achieve this, it is necessary to determine to depict the pixels depending on whether its texel is transparent or not. In rendering, painting is therefore carried out as information on the transparent texel is read. The other procedures are identical to those for the opaque polygons. The processing order of steps (3) to (4) is preferable to its opposite processing order, because it is difficult to read information on the transparent texel in rendering.

(5): The semitransparent polygons are then processed, in which the data of all the pixels that have passed the Z-value comparison are valid. Hence, texture data are read for each pixel and painted with conversion to color data (blending). Since this blending involves the order of depicting polygons as an important matter, the Z-sorting is carried out by any appropriate means in advance.

**[0079]** Referring to Figs.7 to 13, a fragment CG system according to the embodiment of the present invention will now be detailed.

**[0080]** First the operation of rearrangement in the Y-direction will be described. Fig.7 indicates the structures of a Y index buffer and a Y sort buffer. In the Y index buffer, are stored the top addresses of links of polygon lists having fragment lines into each of which a minimum value Ymin of Y coordinates is written. In the Y sort buffer, polygon data are stored in the order along which the data are inputted. As LINK Y parameters in the Y sort buffer, the addresses of the next polygons belonging to the same line are stored.

**[0081]** Accordingly, when it is desired to know polygons that belong to a specified line, it is enough to trace back the links from a corresponding Y INDEX address to a polygon of which LINK Y is END.

**[0082]** An example shown in Fig.7 will be described. When observing the Y index buffer at the uppermost fragment line whose address is zero, its content is "EMPTY," showing that there is no polygon. The same is applied to the second line. But the contents of the Y index buffer at its address 2 showing the third line are "ADR8". Hence, making access to ADR8 of the Y sort buffer leads to a LINK Y content of "ADR5." Hence, the next access is made to ADR5. After this, the similar access is made to ADR3 and ADR1 in sequence. Because the LINK Y content at ADR1 is "END," the link terminates. An identical processing is executed for all the fragment lines.

**[0083]** The operation for storing polygon data will be described.

**[0084]** The assumption is made that in the state shown in Fig.7, polygon data satisfying that Ymin is Y INDEX=5 are inputted. Because the fifth content of the Y index buffer is "EMPTY," the LINK Y of the polygon data becomes "END." This polygon data is stored at ADR10 of the Y sort buffer; also, "ADR10," which is an address of the polygon data that have just been stored, is written at the fifth line of the Y index buffer. This new updated condition is shown in Fig.8.

**[0085]** Next, the operation will be described with a situation that in the state shown in Fig.8, polygon data satisfying that Ymin is Y INDEX=2 are inputted. Since an address of polygon data has already been stored at the second line of the Y index buffer ("ADR8" in Fig.8), such polygon data is to be placed after a polygon to be newly written. Therefore, the LINK Y of the polygon data is set to "ADR8," the value of the original Y INDEX; also, "ADR11," which is the address of the polygon data that have just been stored, is written at the second line of the Y index buffer. This new updated condition is shown in Fig.9.

**[0086]** Fig.10 is an illustration for explaining rearrangement in the X-direction. The rearrangement in the X-direction is processed similarly to that in the Y-direction, in which an Xmin in its line is obtained, then the rearrangement is performed using the obtained value.

**[0087]** As described above, by performing the rearrangements in both the X- and Y-directions, a fragment at which a polygon becomes valid for the first time (that is, the polygon that makes a fragment concerned valid for the first time) is found. For example, as shown in Fig. 11, a fragment of the least X-coordinate value is selected among the least Y-coordinate values in each polygon.

**[0088]** In addition, when referring to the rearranged results, all the lines from zero to a concerned line (row) are regarded as valid. Thus, if a certain polygon is concerned, a region validating the polygon is shown as in Fig.12, resulting in that the valid region for the polygon includes unnecessary fragments. In this case, the link is therefore updated as below.

**[0089]** First, after the completion of the polygon processing, polygons that become invalid in the fragments subsequently processed are removed from the links. In an example shown in Fig.13(a), fragments F1 to F7 are removed from the X LINK after the rendering. The fragment F7 is also removed from the Y LINK after the rendering. A polygon's valid region obtained by performing the link update is exemplified in Fig.13(b).

**[0090]** The following are steps explaining the X-directional link with reference to Fig.13.

(1): Among the intersections made between the polygon's edges and the horizontal boundaries of the fragments extending along the X-direction, an intersection having a larger X-coordinate value (P2) is examined.

(2): Comparison is made between the X-coordinate value at the point examined in the above step (1) and X-coordinate values of boundaries (boundaries 1 to 3) having larger X-coordinate values selected among the boundaries vertical to an X-coordinate of a fragment residing in a line currently concerned.

(3): In cases the comparison result is "small (boundaries 1 and 2)," the polygon undergoes the link processing, validating it toward the next fragment as well.

(4): Where the comparison result is "larger," the polygon is removed from the link after rendering, invalidating it from the processing for the next fragment (boundary 3).

**[0091]** The Y-directional link processing is also performed with respect to the Y-direction by the identical procedures.

**[0092]** As described above, in the embodiment of the present invention, instead of dividing the polygons fragment by fragment, the polygons are rearranged in both the vertical direction and the horizontal direction to a scanning line, thus polygons included in each fragment being able to be retrieved. Since this retrieved information is used and the polygons are divided every fragment now concerned, it is possible to read and process polygons included in each fragment, without producing polygons newly. Thus, unlike the conventional technique, a huge amount of calculation necessary for dividing all the polygons every fragment to be processed can be eliminated and it is not necessary to install a storage with greater amounts of capacity that has been needed to store information about the polygons produced by the division. As a result, the processing time is reduced and it is possible to realize systems in which only a smaller capacity of storage is required. In other words, cost-saving and high-performance systems can be achieved.

**[0093]** Such systems are also attributed to the fact that the processing of the polygons for the semitransparent nature is separated for opaque polygons, polygons with transparent pixels, and semitransparent polygons and the semitransparent polygons are processed at the last stage.

Industrial Applicability:

**[0094]** Thus, in the image processor where a screen is divided into regions at a predetermined size and processing is performed region by region, owing to the fact that image constituting elements are not merely rearranged in a vertical direction to a scanning line but also rearranged in a horizontal direction to the scanning line, then the image processing is performed based on the rearranged image constituting elements, polygons included in each region can be retrieved. Thus, dividing the polygons every region currently concerned, using this retrieved information, makes it possible to read and process polygons included in each region, without producing polygons newly. As a result, a period of processing time is reduced and a memory capacity necessary for the storage can be lowered.

**[0095]** Additionally, as to the image processing for each divided region, objects to be processed are sectioned into types of opaque polygons, polygons with transparent pixels, and semitransparent polygons, then processed in the order of the opaque polygons, the polygons with transparent pixels, and the semitransparent polygons, so that the semitransparent polygons can be processed even when texture data are given the polygons.

## Claims

1. An image processor in which a screen is divided into regions of a predetermined size, and processing on polygon data is performed region by region, comprising first rearranging means for providing constituting elements of a plurality of polygons, falling into individual regions on the screen in a direction vertical to a scanning line, with vertical links mutually linking constituting elements of a polygon, and for rearranging links to constituting elements falling into the same region, thus linking these elements;

   second rearranging means for providing constituting elements of a plurality of polygons, falling into individual regions on the screen in the direction of the scanning line, with horizontal links mutually linking constituting elements of a polygon, and for rearranging links to constituting elements falling into the same region, thus linking these elements; and

   an image processing means for

   retrieving and processing constituting elements of diverse polygons falling in a given region, the sequence of constituting elements being given by rearranged links.

2. An image processor according to claim 1, wherein

   the first rearranging means is operable to perform the rearrangement based on minimum or maximum values of the constituting elements with reference to the vertical direction; and

   the second rearranging means is operable to perform the rearrangement based on minimum or maximum values of the constituting elements with reference to the horizontal direction, in vertical coordinates of a region to be processed by the first rearranging means.

3. An image processor according to claim 1 or claim 2, wherein the first and second rearranging means determine whether or not, after completion of the processing of a polygon in the image processing means for a said region, the constituting elements for said polygon to be used by the image processing means are included in a region to be image-processed at the next processing, and perform link update processing which removes from the link the constituting elements which are determined to not be included.

4. An image processor according to any preceding claim, wherein the image processing means, as to image processing for each divided region, not only sections polygons to be processed into opaque polygons, polygons with transparent pixels, and semitransparent polygons but also processes in the order of the opaque polygons, polygons with transparent pixels, and semitransparent polygons.

5. An image processing method by which a screen is divided into regions of a predetermined size, and processing on polygon data is performed region by region, the method comprising the steps of:

   a first rearranging step of providing constituting elements of a plurality of polygons, falling into individual regions on the screen in a direction vertical to a scanning line, with vertical links mutually linking constituting elements of a polygon, and rearranging links to constituting elements falling into the same region, thus linking these elements;

   a second rearranging step of

   for providing constituting elements of a plurality of polygons, falling into individual regions on the screen in the direction of the scanning line, with horizontal links mutually linking constituting elements of a polygon, and rearranging links to constituting elements falling into the same region, thus linking these elements; and

   an image processing step of

   retrieving and processing constituting elements of diverse polygons falling in a given region, the sequence of constituting elements being given by rearranged links.

6. A method according to claim 5, wherein,

   in the first rearranging step, the rearrangement is performed based on minimum or maximum

values of the constituting elements with reference to the vertical direction, and

in the second rearranging step, the rearrangement is performed based on minimum or maximum values of the constituting elements with reference to the horizontal direction, in vertical coordinates of a region to be processed in the first rearranging step.

**7.** A method according to claim 5 or claim 6, wherein in the first and second rearranging steps, link update processing is performed to invalidate an unnecessary region among the regions corresponding to the constituting elements.

**8.** A method according to any one of claims 5 to 7, wherein in the image processing step, the image processing for each divided region is performed with polygons to be processed sectioned into opaque polygons, polygons with transparent pixels, and semitransparent polygons, and performed in the order of the opaque polygons, polygons with transparent pixels, and semitransparent polygons.

**9.** An image processor according to any of claims 1 to 4, including:

a memory buffer arranged to memorize said constituting elements and links; and
an index buffer arranged to memorize, for every divided region, a location of memorization of at least one of the constituting elements among the constituting elements included in that region; and wherein
said image processing means reads from the memory buffer by reference to the index buffer constituting elements included in regions at which the processing is to be performed and performs desired image processing.

**10.** An image processor according to claim 9, wherein the index buffer has a vertical-position index buffer memorizing, for every divided region in the direction vertical to the scanning line, a location of memorization of at least one of the image constituting elements among the constituting elements included in said divided regions at the positions in said vertical direction and a horizontal-position index buffer memorizing, for every divided region in the direction horizontal to the scanning line, a location of memorization of at least one of the image constituting elements among the constituting elements included in said divided regions at the positions in said horizontal direction.

**Patentansprüche**

**1.** Bildprozessor, bei dem ein Bildschirm in Bereiche einer vorbestimmten Größe unterteilt ist, und die Verarbeitung von Daten eines Vielecks Bereich für Bereich durchgeführt wird, umfassend erste Reorganisationseinrichtungen, um Bestandteile einer Vielzahl von Vielecken zur Verfügung zu stellen, welche auf dem Bildschirm in einer vertikalen Richtung im Verhältnis zu einer Abtastzeile zu einzelnen Bereichen gehören, wobei vertikale Verknüpfungen Bestandteile eines Vielecks gegenseitig verknüpfen, und um Verknüpfungen mit Bestandteilen, welche zu demselben Bereich gehören, zu reorganisieren, wodurch diese Teile verknüpft werden;
zweite Reorganisationseinrichtungen, um Bestandteile einer Vielzahl von Vielecken zur Verfügung zu stellen, welche auf dem Bildschirm in der Richtung der Abtastzeile zu einzelnen Bereichen gehören, wobei horizontale Verknüpfungen Bestandteile eines Vielecks gegenseitig verknüpfen, und um Verknüpfungen mit Bestandteilen, welche zu demselben Bereich gehören, zu reorganisieren, wodurch diese Teile verknüpft werden; und
eine Bildverarbeitungseinrichtung zum
Abrufen und Verarbeiten von Bestandteilen verschiedener Vielecke, welche zu einem gegebenen Bereich gehören, wobei die Reihenfolge von Bestandteilen durch die reorganisierten Verknüpfungen vorgegeben wird.

**2.** Bildprozessor nach Anspruch 1, wobei
die ersten Reorganisationseinrichtungen so funktionieren, dass sie die Reorganisation auf der Grundlage von Minimum- oder Maximumwerten der Bestandteile im Verhältnis zu der vertikalen Richtung durchführen; und
die zweiten Reorganisationseinrichtungen so funktionieren, dass sie die Reorganisation auf der Grundlage von Minimum- oder Maximumwerten der Bestandteile im Verhältnis zu der horizontalen Richtung, in vertikalen Koordinaten eines Bereiches durchführen, der durch die ersten Reorganisationseinrichtungen verarbeitet werden soll.

**3.** Bildprozessor nach Anspruch 1 oder Anspruch 2, wobei die ersten und zweiten Reorganisationseinrichtungen, nachdem die Verarbeitung eines Vielecks in der Bildverarbeitungseinrichtung für den Bereich abgeschlossen ist, bestimmen, ob die Bestandteile für das Vieleck, das durch die Bildverarbeitungseinrichtung verwendet werden soll, in einem Bereich eingeschlossen sind oder nicht, welcher beim nächsten Verarbeiten bildverarbeitet werden soll, und eine Verarbeitung zur Verknüpfungsaktualisierung durchführen, bei welcher die Bestandteile aus der Verknüpfung entfernt werden, die bestimmt werden, nicht eingeschlossen zu wer-

den.

**4.** Bildprozessor nach irgendeinem der vorhergehenden Ansprüche, wobei die Bildverarbeitungseinrichtung, zum Bildverarbeiten für jeden unterteilten Bereich, Vielecke, welche verarbeitet werden sollen, nicht nur in undurchsichtige Vielecke, Vielecke mit durchsichtigen Pixeln, und halbdurchsichtige Vielecke aufgeteilt, sondern auch in der Reihenfolge undurchsichtige Vielecke, Vielecke mit durchsichtigen Pixeln und halbdurchsichtige Vielecke verarbeitet.

**5.** Bildverarbeitungsverfahren, bei dem ein Bildschirm in Bereiche einer vorbestimmten Größe unterteilt ist, und die Verarbeitung von Daten eines Vielecks Bereich für Bereich durchgeführt wird, wobei das Verfahren die folgenden Schritte umfasst:

einen ersten Reorganisationsschritt, um Bestandteile einer Vielzahl von Vielecken zur Verfügung zu stellen, welche auf dem Bildschirm in einer vertikalen Richtung im Verhältnis zu einer Abtastzeile zu einzelnen Bereichen gehören, wobei vertikale Verknüpfungen Bestandteile eines Vielecks gegenseitig verknüpfen, und um Verknüpfungen mit Bestandteilen reorganisieren, welche zu demselben. Bereich gehören, wodurch diese Teile verknüpft werden;

einen zweiten Reorganisationsschritt, um Bestandteile einer Vielzahl von Vielecken zur Verfügung zu stellen, welche auf dem Bildschirm in der Richtung der Abtastzeile zu einzelnen Bereichen gehören, wobei horizontale Verknüpfungen Bestandteile eines Vielecks gegenseitig verknüpfen, und um Verknüpfungen mit Bestandteilen reorganisieren, welche zu demselben Bereich gehören, wodurch diese Teile verknüpft werden; und

einen Bildverarbeitungsschritt zum

Abrufen und Verarbeiten von Bestandteilen von verschiedenen Vielecken, welche zu einem gegebenen Bereich gehören, wobei die Reihenfolge von Bestandteilen durch die reorganisierten Verknüpfungen vorgegeben wird.

**6.** Verfahren nach Anspruch 5, wobei
in dem ersten Reorganisationsschritt die Reorganisation auf der Grundlage von Minimum- oder Maximumwerten der Bestandteile im Verhältnis zur vertikalen Richtung durchgeführt wird; und
in dem zweiten Reorganisationsschritt die Reorganisation auf der Grundlage von Minimum- oder Maximumwerten der Bestandteile im Verhältnis zur horizontalen Richtung, in vertikalen Koordinaten eines Bereiches durchgeführt wird, der in dem ersten Reorganisationsschritt verarbeitet werden soll.

**7.** Verfahren nach Anspruch 5 oder Anspruch 6, wobei in dem ersten und zweiten Reorganisationsschritt die Verarbeitung zur Verknüpfungsaktualisierung durchgeführt wird, um einen nicht notwendigen Bereich unter den Bereichen zu annullieren, welche den Bestandteilen entsprechen.

**8.** Verfahren nach irgendeinem der Ansprüche 5 bis 7, wobei in dem Bildverarbeitungsschritt die Bildverarbeitung für jeden unterteilten Bereich mit Vielecken durchgeführt wird, die in undurchsichtige Vielecke, Vielecke mit durchsichtigen Pixeln und halbdurchsichtige Vielecke aufgeteilt verarbeitet werden sollen, und in der Reihenfolge undurchsichtige Vielecke, Vielecke mit durchsichtigen Pixeln und halbdurchsichtige Vielecke durchgeführt wird.

**9.** Bildprozessor nach irgendeinem der Ansprüche 1 bis 4, einschließend:

einen Speicherpuffer, welcher angeordnet ist, um die Bestandteile und Verknüpfungen zu speichern; und

einen Indexpuffer, welcher angeordnet ist, um, für jeden unterteilten Bereich, eine Speicherstelle von mindestens einem der Bestandteile unter den Bestandteilen zu speichern, welche in diesem Bereich eingeschlossen sind; und wobei

die Bildverarbeitungseinrichtung unter Bezugnahme auf den Indexpuffer Bestandteile aus dem Speicherpuffer liest, die in Bereichen eingeschlossen sind, an denen die Verarbeitung durchgeführt werden soll, und die gewünschte Bildverarbeitung durchführt.

**10.** Bildprozessor nach Anspruch 9, wobei der Indexpuffer einen Indexpuffer für eine vertikale Position aufweist, welcher, für jeden unterteilten Bereich in vertikaler Richtung zu der Abtastzeile, eine Speicherstelle von mindestens einem der Bildbestandteile unter den Bestandteilen speichert, die in den unterteilten Bereichen an den Positionen in der vertikalen Richtung eingeschlossen sind, und einen Indexpuffer für eine horizontale Position, welcher, für jeden unterteilten Bereich in horizontaler Richtung zu der Abtastzeile, eine Speicherstelle von mindestens einem der Bildbestandteile unter den Bestandteilen speichert, die in den unterteilten Bereichen an den Positionen in der horizontalen Richtung eingeschlossen sind.

**Revendications**

1. Processeur d'image dans lequel un écran est divisé en régions d'une taille prédéterminée, et un traitement de données de polygones est exécuté région par région, comprenant des premiers moyens de réorganisation pour fournir des éléments de constitution d'une pluralité de polygones, venant se placer dans des régions individuelles à l'écran dans une direction verticale à une ligne d'exploration, avec des liaisons verticales reliant mutuellement les éléments de constitution d'un polygone, et pour réorganiser les liaisons aux éléments de constitution venant se placer dans la même région, reliant ainsi ces éléments ;

   des seconds moyens de réorganisation pour fournir des éléments de constitution d'une pluralité de polygones, venant se placer dans des régions individuelles à l'écran dans la direction de la ligne d'exploration, avec des liaisons horizontales reliant mutuellement des éléments de constitution d'un polygone, et pour réorganiser les liaisons aux éléments de constitution venant se placer dans la même région, reliant ainsi ces éléments ; et

   des moyens de traitement d'image pour

   récupérer et traiter des éléments de constitution de divers polygones venant se placer dans une région donnée, la séquence d'éléments de constitution étant fournie par les liaisons réorganisées.

2. Processeur d'image selon la revendication 1, dans lequel

   les premiers moyens de réorganisation peuvent fonctionner pour exécuter la réorganisation en fonction de valeurs minimum ou maximum des éléments de constitution par rapport à la direction verticale ; et

   les seconds moyens de réorganisation peuvent fonctionner pour exécuter la réorganisation en fonction de valeurs minimum ou maximum des éléments de constitution par rapport à la direction horizontale, dans des coordonnées verticales d'une région à traiter par les premiers moyens de réorganisation.

3. Processeur d'image selon la revendication 1 ou 2, dans lequel les premiers et seconds moyens de réorganisation déterminent, une fois le traitement d'un polygone dans les moyens de traitement d'image pour une dite région terminé, si les éléments de constitution dudit polygone à utiliser par les moyens de traitement d'image sont compris ou non dans une région à laquelle appliquer un traitement d'image lors du traitement suivant, et exécutent un traitement de mise à jour de liaison qui supprime de la liaison les éléments de constitution qui sont déterminés comme ne devant pas être compris.

4. Processeur d'image selon l'une quelconque des revendications précédentes, dans lequel les moyens de traitement d'image, comme pour le traitement d'image pour chaque région séparée, ne découpent pas seulement des polygones à traiter en des polygones opaques, des polygones avec des pixels transparents, et des polygones semi-transparents mais traitent également dans l'ordre des polygones opaques, des polygones avec des pixels transparents, puis des polygones semi-transparents.

5. Procédé de traitement d'image grâce auquel un écran est divisé en régions d'une taille prédéterminée, et un traitement de données de polygone est exécuté région par région, le procédé comprenant les étapes suivantes :

   une première étape de réorganisation pour fournir des éléments de constitution d'une pluralité de polygones, venant se placer dans des régions individuelles à l'écran dans une direction verticale à une ligne d'exploration, avec des liaisons verticales reliant mutuellement des éléments de constitution d'un polygone, et des liaisons de réorganisation aux éléments de constitution venant se placer dans la même région, reliant ainsi ces éléments ;

   une seconde étape de réorganisation pour fournir des éléments de constitution d'une pluralité de polygones, venant se placer dans des régions individuelles à l'écran dans la direction de la ligne d'exploration, avec des liaisons horizontales reliant mutuellement des éléments de constitution d'un polygone, et des liaisons de réorganisation aux éléments de constitution venant se placer dans la même région, reliant ainsi ces éléments ; et

   une étape de traitement d'image pour

   récupérer et traiter des éléments de constitution de divers polygones venant se placer dans une région donnée, la séquence d'éléments de constitution étant fournie par les liaisons réorganisées.

6. Procédé selon la revendication 5, dans lequel,

   dans la première étape de réorganisation, la réorganisation est exécutée en fonction de valeurs minimum ou maximum des éléments de constitution par rapport à la direction verticale ; et

   dans la seconde étape de réorganisation, la réorganisation est exécutée en fonction de valeurs minimum ou maximum des éléments de constitution par rapport à la direction horizontale, dans des coordonnées verticales d'une région à traiter lors de la première étape de réorganisation.

7. Procédé selon la revendication 5 ou 6, dans lequel lors des première et seconde étapes de réorgani-

sation, un traitement de mise à jour de liaison est exécuté pour invalider une région inutile parmi les régions correspondant aux éléments de constitution.

**8.** Procédé selon l'une quelconque des revendications 5 à 7, dans lequel, lors de l'étape de traitement d'image, le traitement d'image pour chaque région séparée est exécuté avec des polygones à traiter découpés en des polygones opaques, des polygones avec des pixels transparents, et des polygones semi-transparents, et est exécuté dans l'ordre des polygones opaques, des polygones avec des pixels transparents, puis des polygones semi-transparents.

**9.** Processeur d'image selon l'une quelconque des revendications 1 à 4, comprenant :

une mémoire tampon agencée pour mémoriser lesdits éléments de constitution et les liaisons ; et
un index tampon agencé pour mémoriser, pour chaque région séparée, un emplacement de mémorisation d'au moins un des éléments de constitution parmi les éléments de constitution compris dans cette région ; et dans lequel
lesdits moyens de traitement d'image consultent dans la mémoire tampon. en référence à l'index tampon, les éléments de constitution compris dans les régions dans lesquelles le traitement doit être exécuté, et exécutent le traitement d'image souhaité.

**10.** Processeur d'image selon la revendication 9, dans lequel l'index tampon comprend un index tampon de position verticale mémorisant, pour chaque région séparée dans la direction verticale par rapport à la ligne d'exploration, un emplacement de mémorisation d'au moins un des éléments de constitution parmi les éléments de constitution compris dans lesdites régions séparées aux positions dans ladite direction verticale, et un index tampon de position horizontale mémorisant, pour chaque région séparée dans la direction horizontale par rapport à la ligne d'exploration, un emplacement de mémorisation d'au moins un des éléments de constitution parmi les éléments de constitution compris dans lesdites régions séparées aux positions dans ladite direction horizontale.

FIG.1

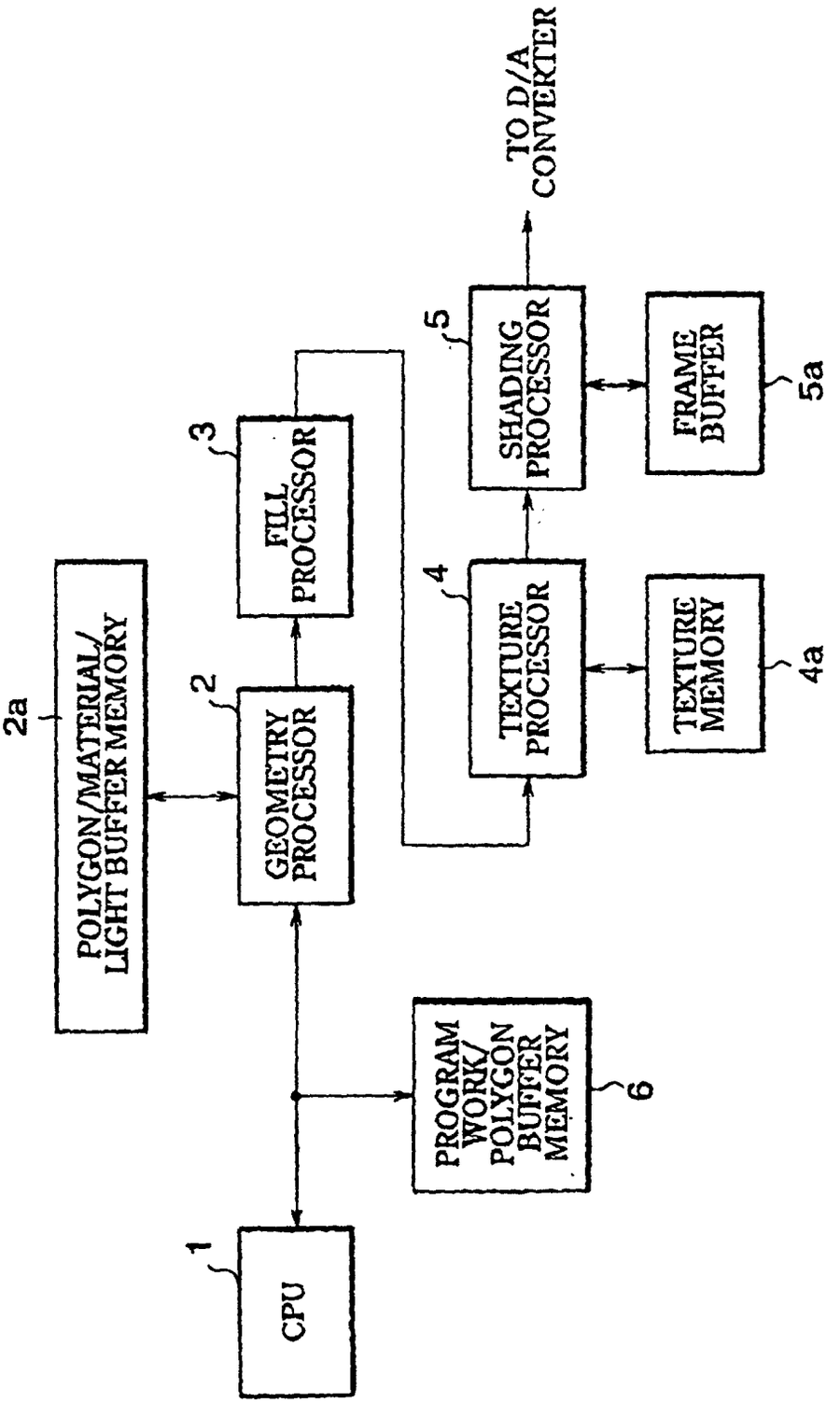

CPU
1
PROGRAM WORK/ POLYGON BUFFER MEMORY
6
POLYGON/MATERIAL/ LIGHT BUFFER MEMORY
2a
GEOMETRY PROCESSOR
2
FILL PROCESSOR
3
TEXTURE PROCESSOR
4
TEXTURE MEMORY
4a
SHADING PROCESSOR
5
FRAME BUFFER
5a
TO D/A CONVERTER

FIG.2

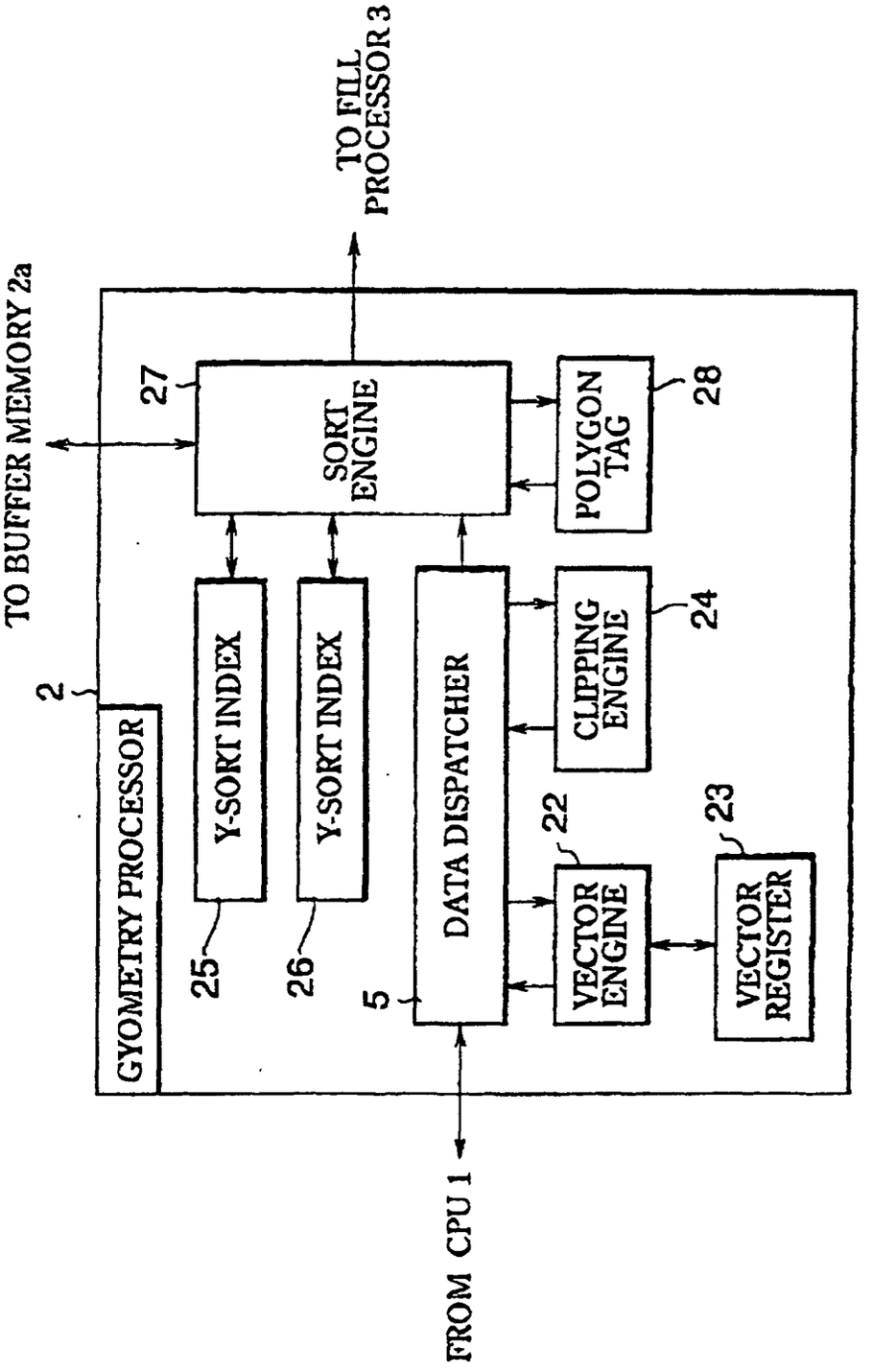

TO FILL PROCESSOR 3
TO BUFFER MEMORY 2a
27
SORT ENGINE
POLYGON TAG
28
2
GYOMETRY PROCESSOR
Y-SORT INDEX
Y-SORT INDEX
DATA DISPATCHER
CLIPPING ENGINE
24
22
VECTOR ENGINE
23
VECTOR REGISTER
25
26
5
FROM CPU 1

FIG.3

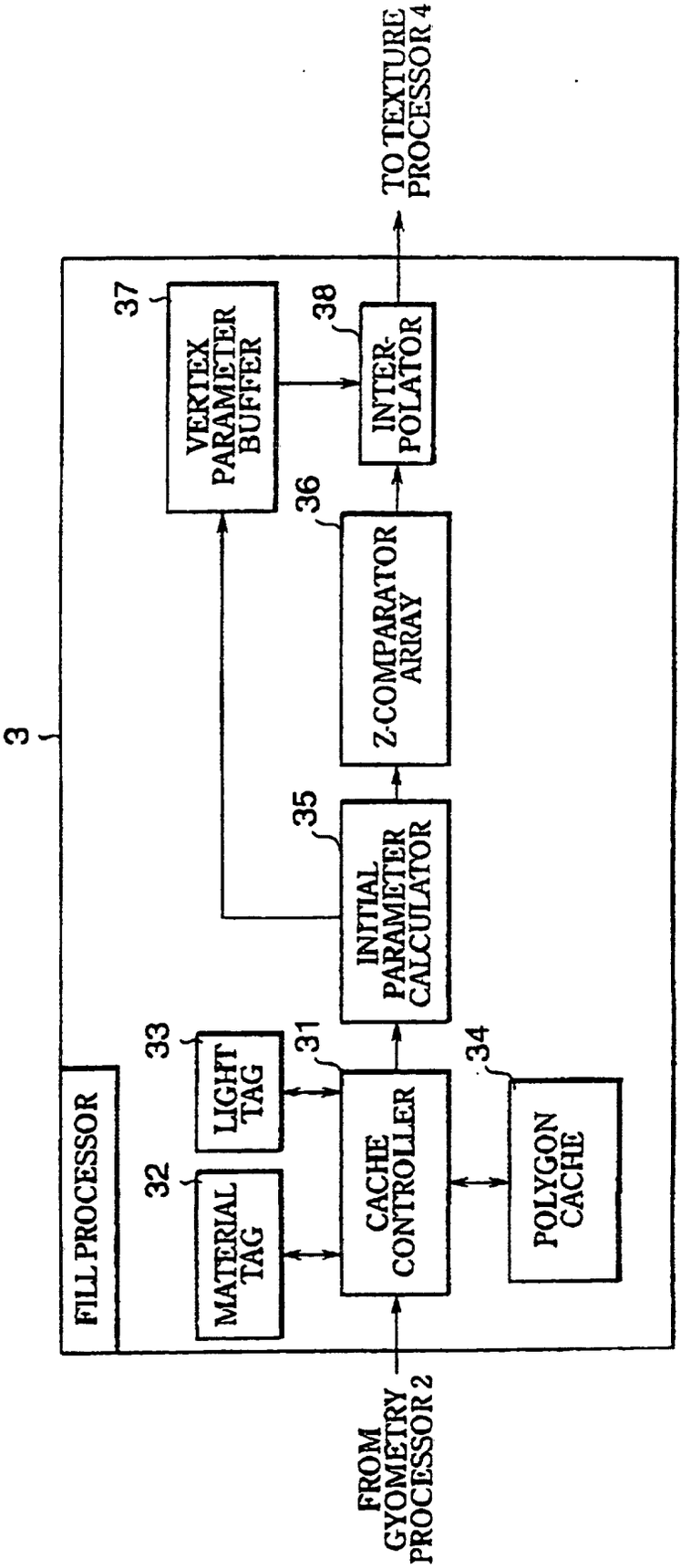

FROM GYOMETRY PROCESSOR 2
TO TEXTURE PROCESSOR 4
3
FILL PROCESSOR
MATERIAL TAG
32
LIGHT TAG
33
CACHE CONTROLLER
31
POLYGON CACHE
34
INITIAL PARAMETER CALCULATOR
35
Z-COMPARATOR ARRAY
36
VERTIEX PARAMETER BUFFER
37
INTER-POLATOR
38

FIG.4

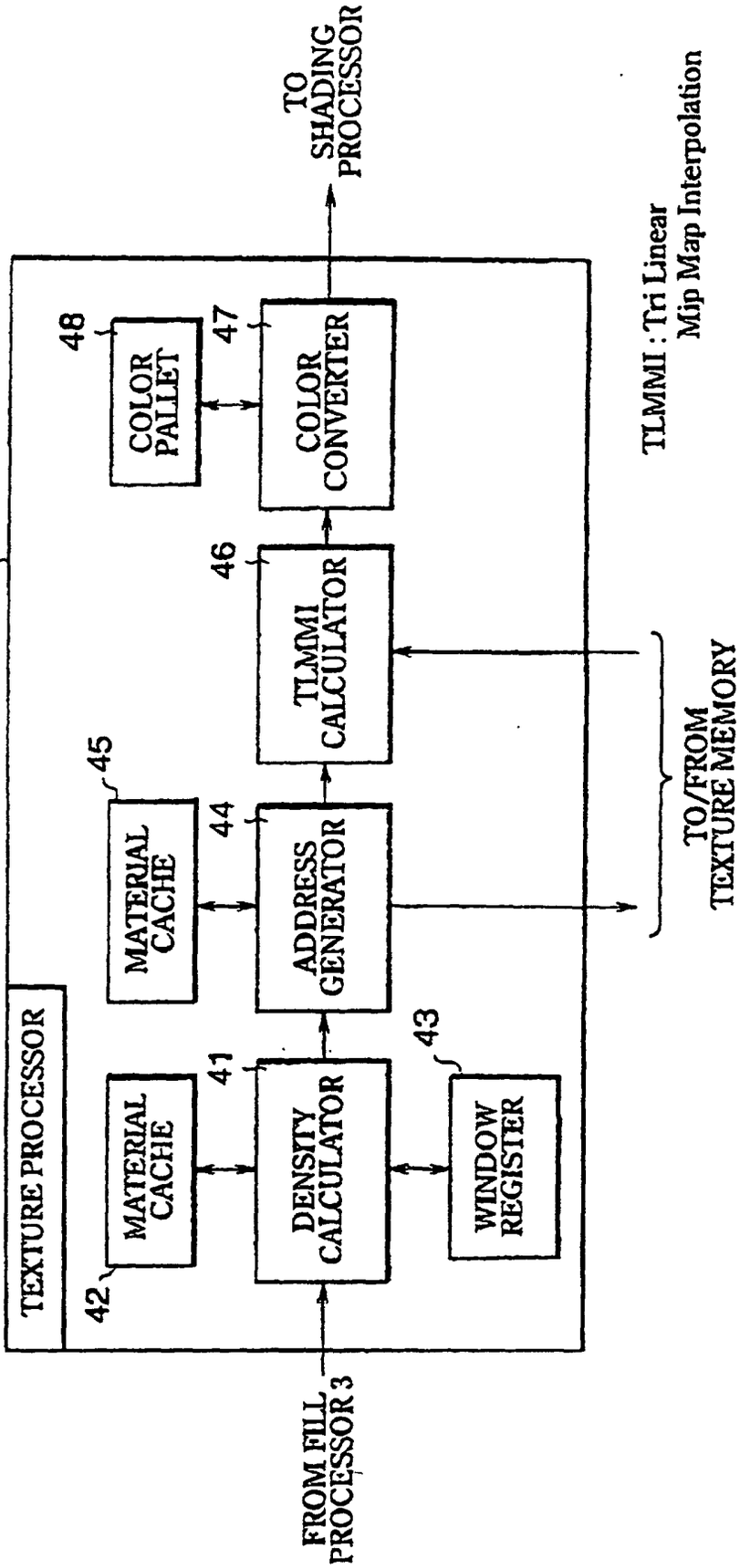
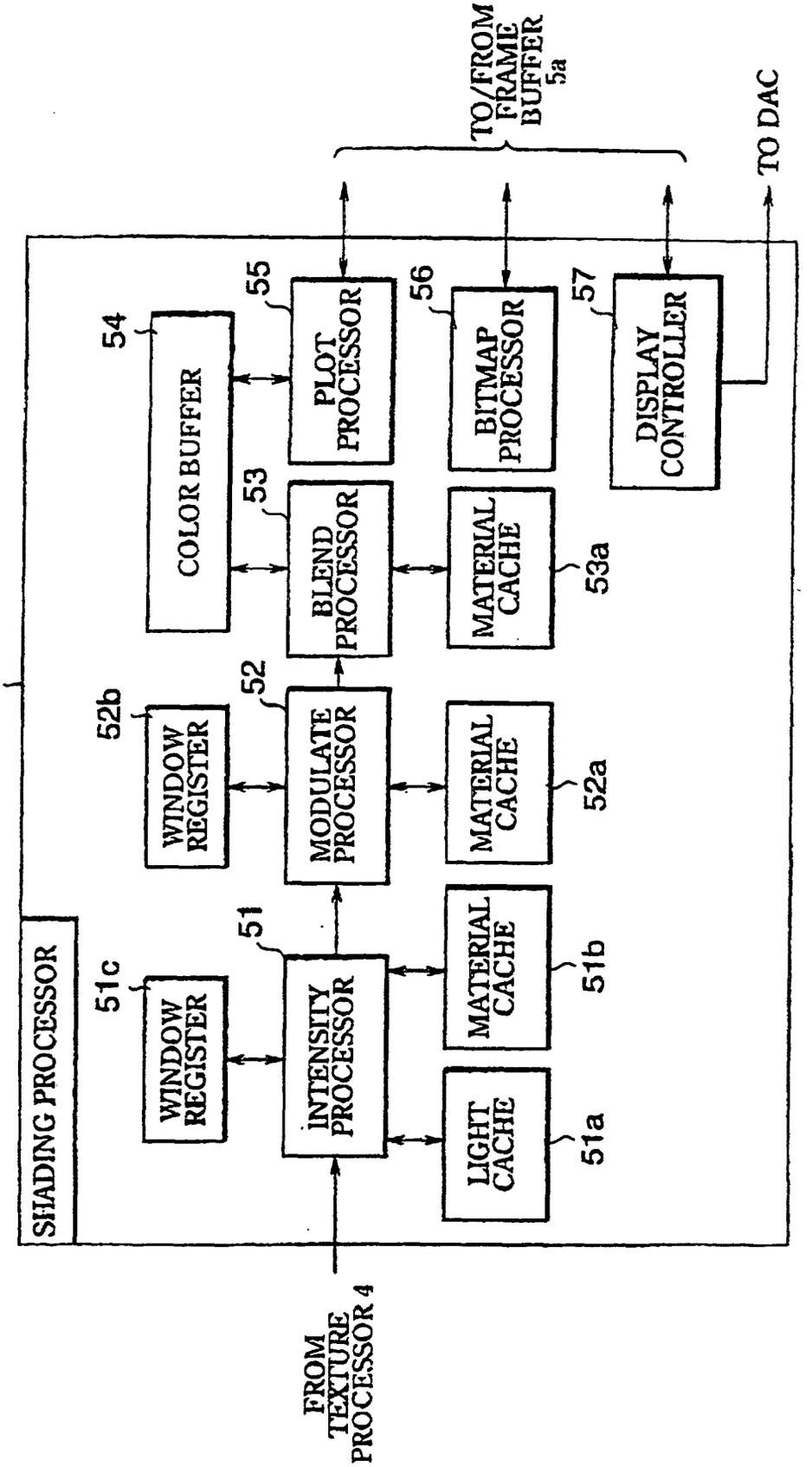

TEXTURE PROCESSOR
4
MATERIAL CACHE
42
DENSITY CALCULATOR
41
WINDOW REGISTER
43
MATERIAL CACHE
45
ADDRESS GENERATOR
44
TLMMI CALCULATOR
46
COLOR PALLET
48
COLOR CONVERTER
47
FROM FILL PROCESSOR 3
TO/FROM TEXTURE MEMORY
TO SHADING PROCESSOR
TLMMI : Tri Linear Mip Map Interpolation

FIG.5


5
SHADING PROCESSOR
FROM TEXTURE PROCESSOR 4
INTENSITY PROCESSOR
51
WINDOW REGISTER
51c
LIGHT CACHE
51a
MATERIAL CACHE
51b
MODULATE PROCESSOR
52
WINDOW REGISTER
52b
MATERIAL CACHE
52a
BLEND PROCESSOR
53
MATERIAL CACHE
53a
COLOR BUFFER
54
PLOT PROCESSOR
55
BITMAP PROCESSOR
56
DISPLAY CONTROLLER
57
TO/FROM FRAME BUFFER 5a
TO DAC

# FIG.6

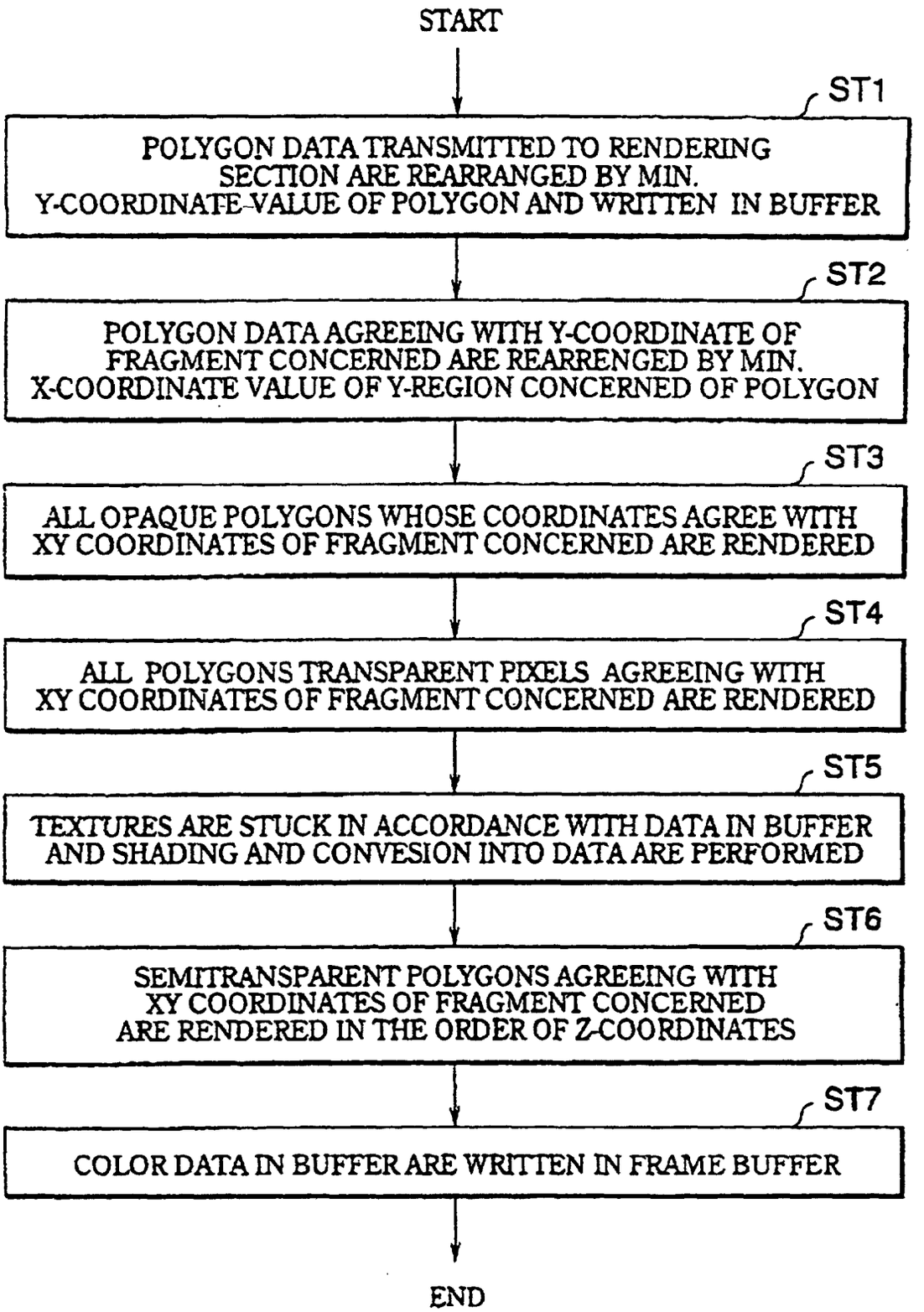

START
ST1
POLYGON DATA TRANSMITTED TO RENDERING SECTION ARE REARRANGED BY MIN. Y-COORDINATE-VALUE OF POLYGON AND WRITTEN IN BUFFER
ST2
POLYGON DATA AGREEING WITH Y-COORDINATE OF FRAGMENT CONCERNED ARE REARRENGED BY MIN. X-COORDINATE VALUE OF Y-REGION CONCERNED OF POLYGON
ST3
ALL OPAQUE POLYGONS WHOSE COORDINATES AGREE WITH XY COORDINATES OF FRAGMENT CONCERNED ARE RENDERED
ST4
ALL POLYGONS TRANSPARENT PIXELS AGREEING WITH XY COORDINATES OF FRAGMENT CONCERNED ARE RENDERED
ST5
TEXTURES ARE STUCK IN ACCORDANCE WITH DATA IN BUFFER AND SHADING AND CONVESION INTO DATA ARE PERFORMED
ST6
SEMITRANSPARENT POLYGONS AGREEING WITH XY COORDINATES OF FRAGMENT CONCERNED ARE RENDERED IN THE ORDER OF Z-COORDINATES
ST7
COLOR DATA IN BUFFER ARE WRITTEN IN FRAME BUFFER
END

FIG.7

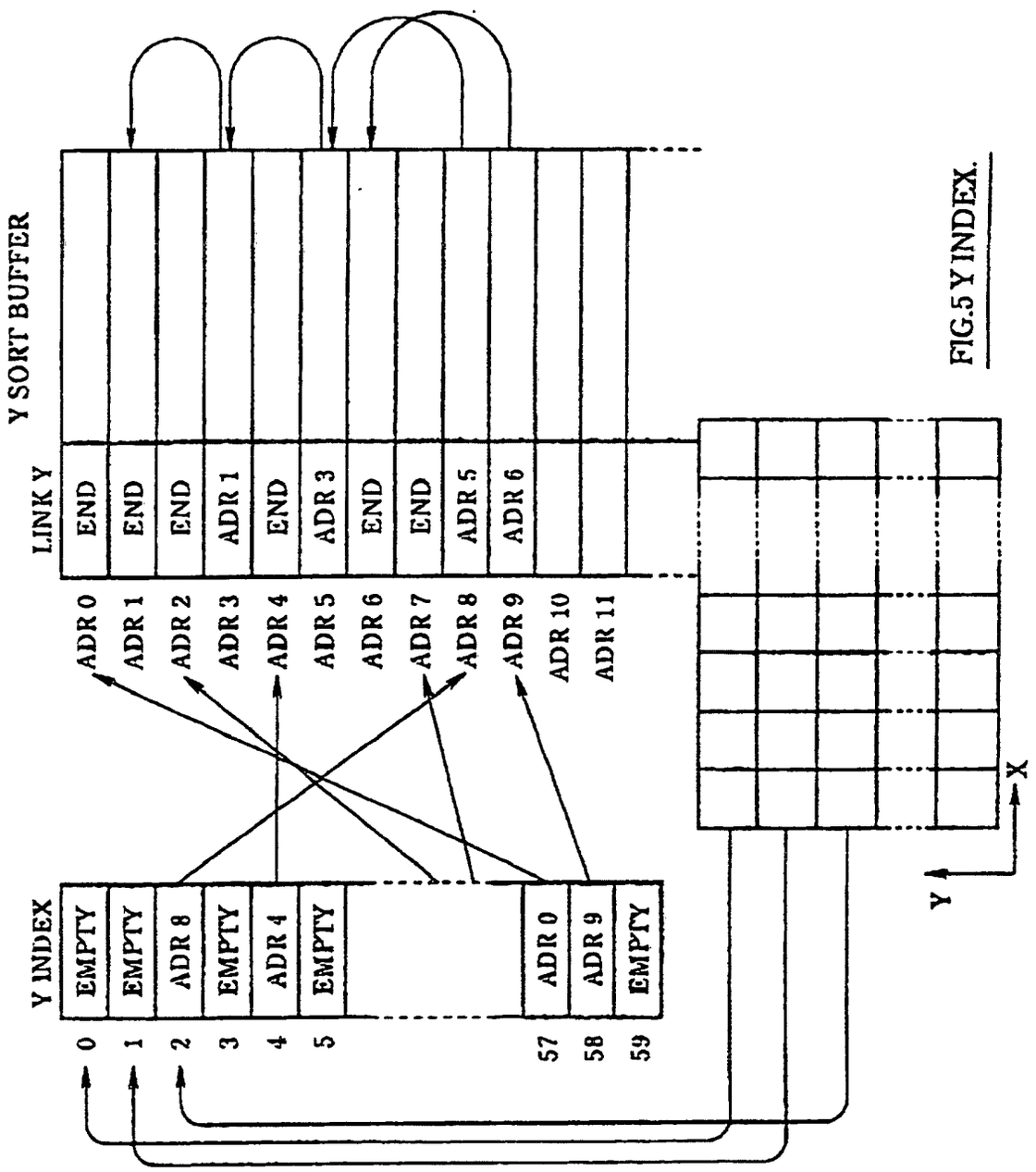



FIG.5 Y INDEX.

FIG.8

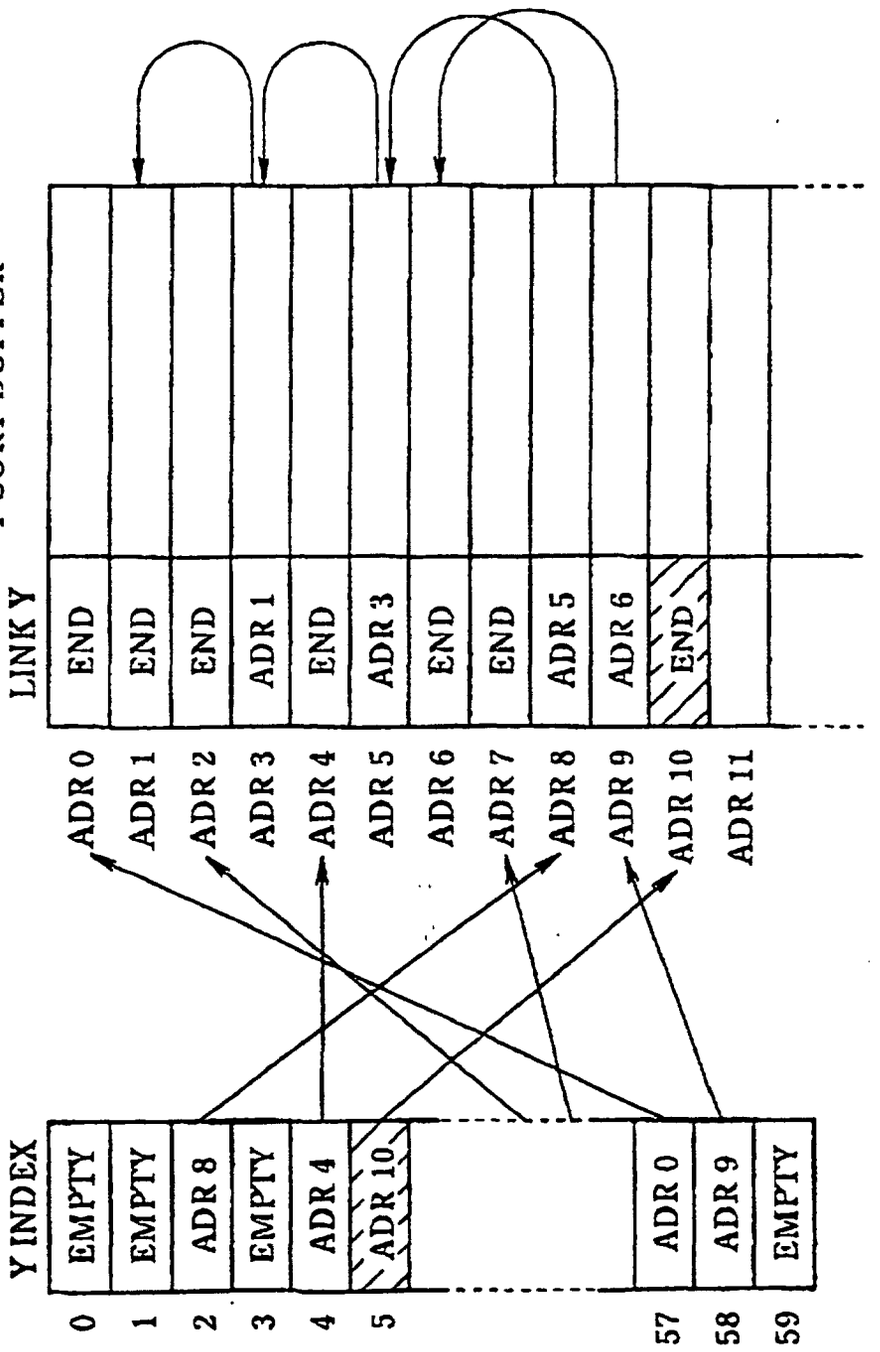

Y SORT BUFFER
LINK Y
END
END
END
ADR 1
END
ADR 3
END
END
ADR 5
ADR 6
END
ADR 0
ADR 1
ADR 2
ADR 3
ADR 4
ADR 5
ADR 6
ADR 7
ADR 8
ADR 9
ADR 10
ADR 11
Y INDEX
EMPTY
EMPTY
ADR 8
EMPTY
ADR 4
ADR 10
ADR 0
ADR 9
EMPTY
0
1
2
3
4
5
57
58
59

FIG.9

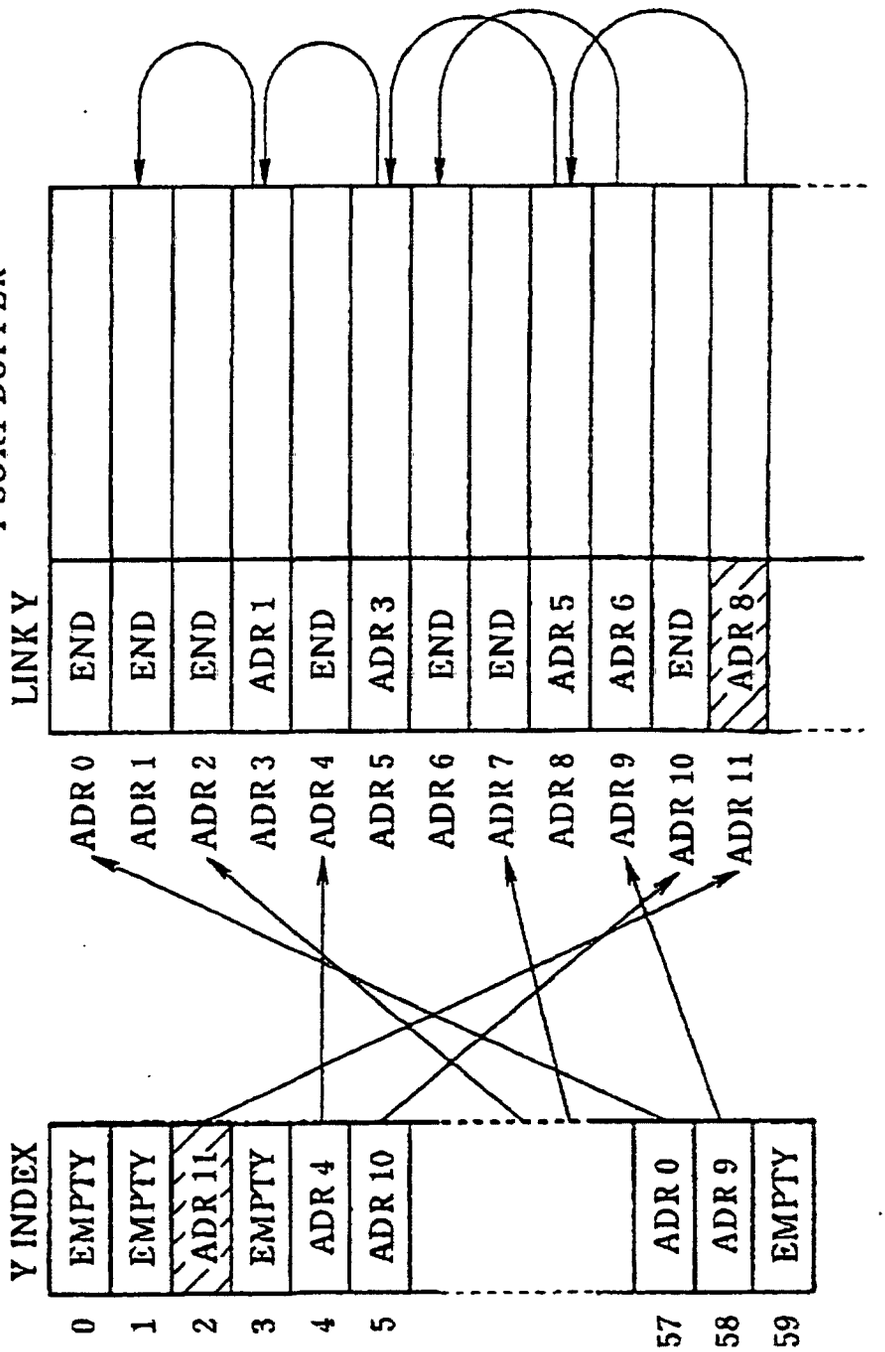

Y SORT BUFFER
LINK Y
END
END
END
ADR 1
END
ADR 3
END
END
ADR 5
ADR 6
END
ADR 8
ADR 0
ADR 1
ADR 2
ADR 3
ADR 4
ADR 5
ADR 6
ADR 7
ADR 8
ADR 9
ADR 10
ADR 11
Y INDEX
EMPTY
EMPTY
ADR 11
EMPTY
ADR 4
ADR 10
ADR 0
ADR 9
EMPTY
0
1
2
3
4
5
57
58
59

FIG.10

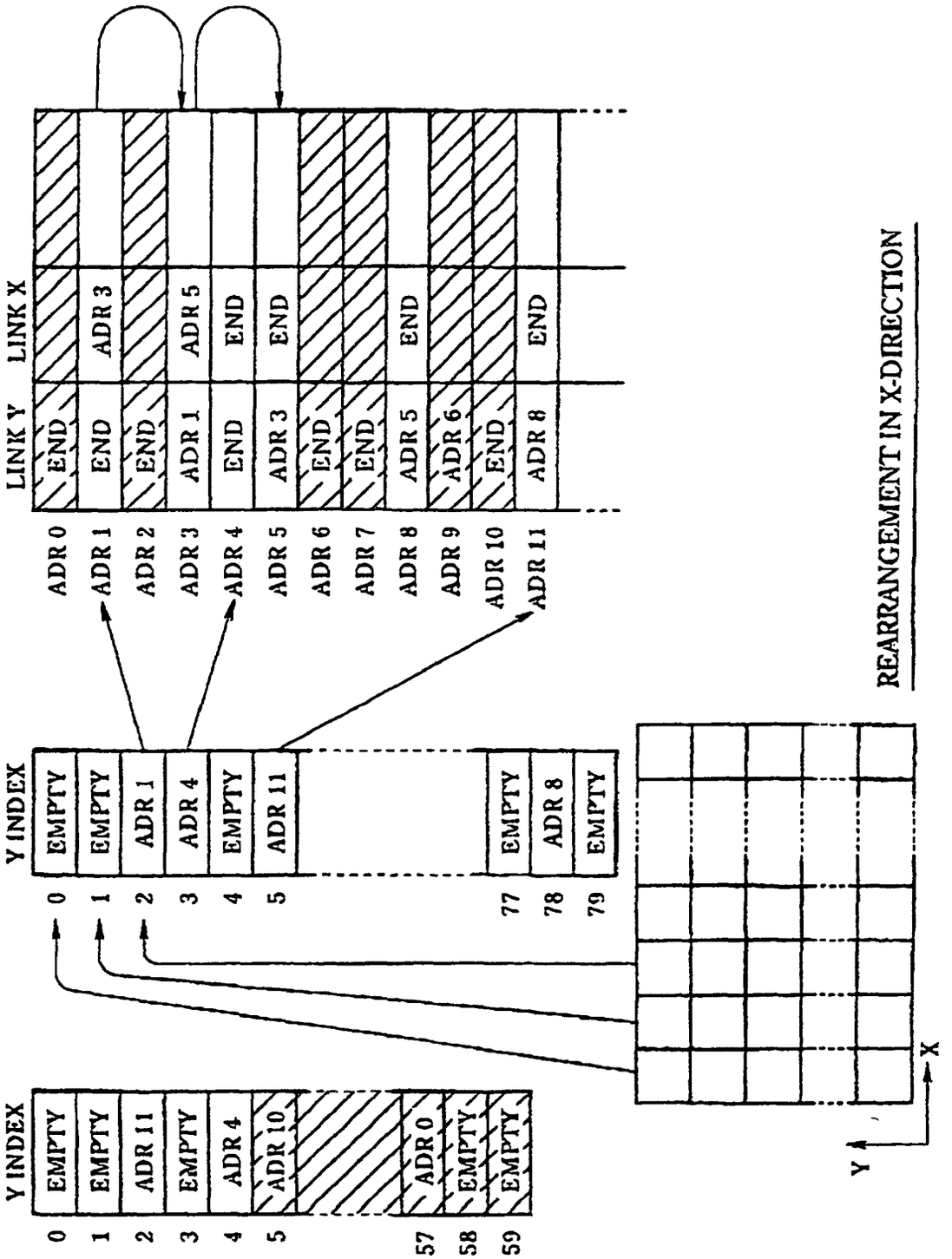

LINK Y
LINK X
END
ADR 1
ADR 3
ADR 5
ADR 6
ADR 8
ADR 0
ADR 2
ADR 4
ADR 7
ADR 9
ADR 10
ADR 11
Y INDEX
EMPTY
0
1
2
3
4
5
57
58
59
77
78
79
REARRANGEMENT IN X-DIRECTION
X
Y

FIG.11

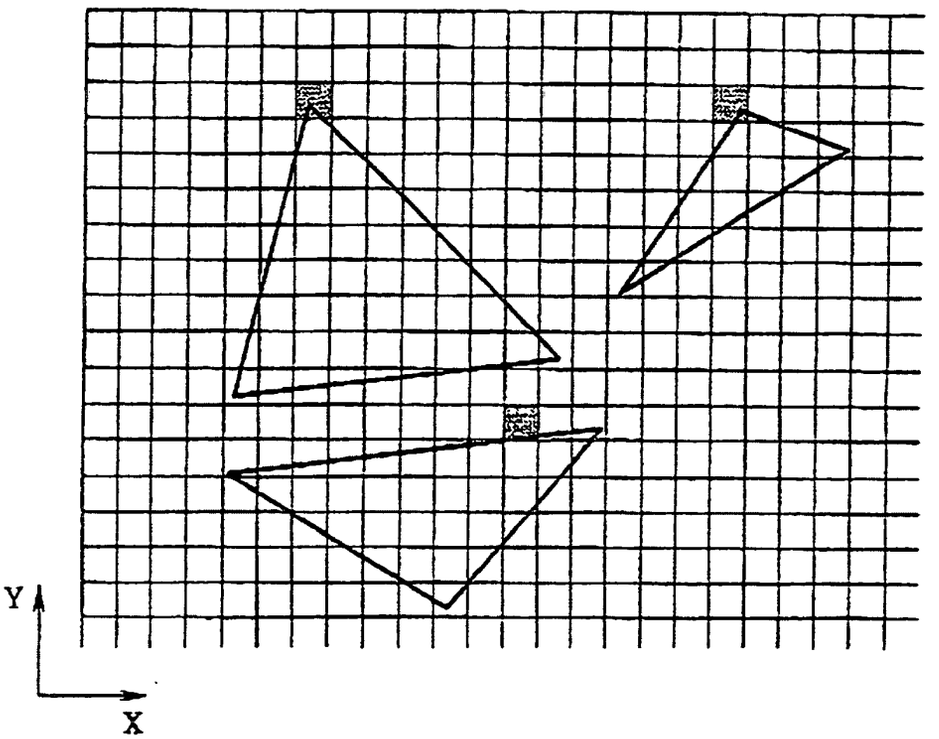
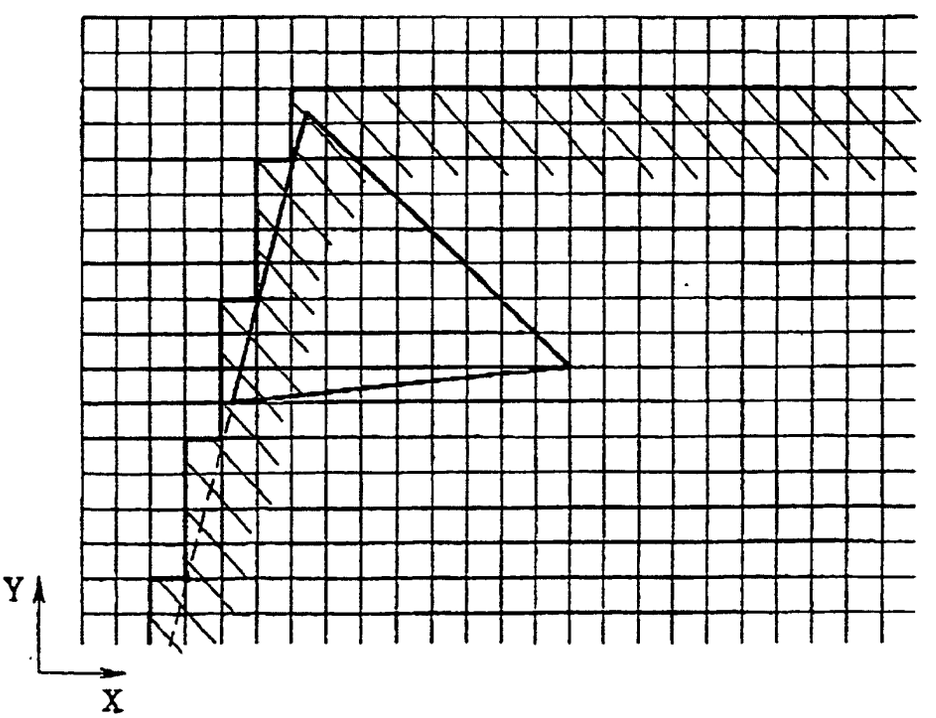
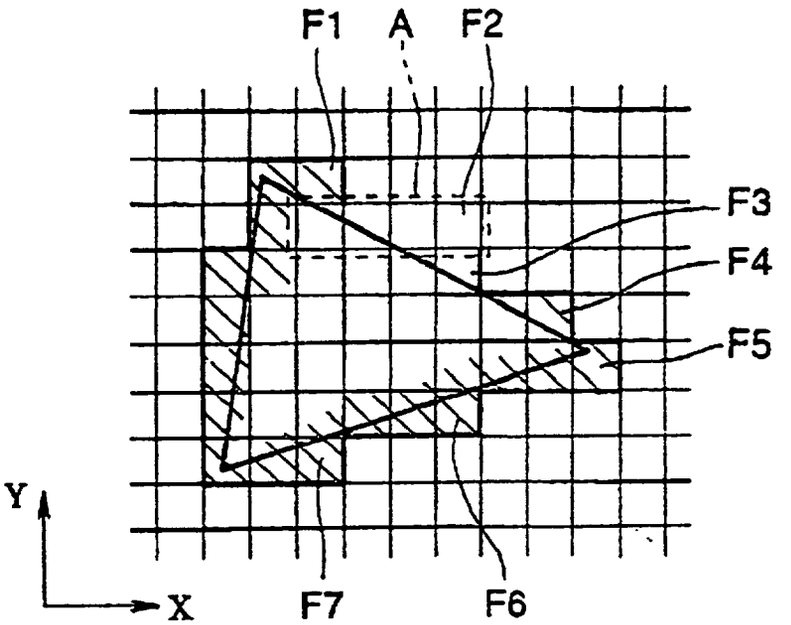


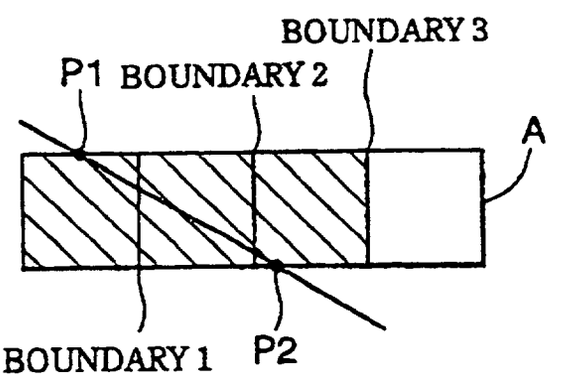

FIG.12



# FIG.13




(a)




(b)

FIG.14

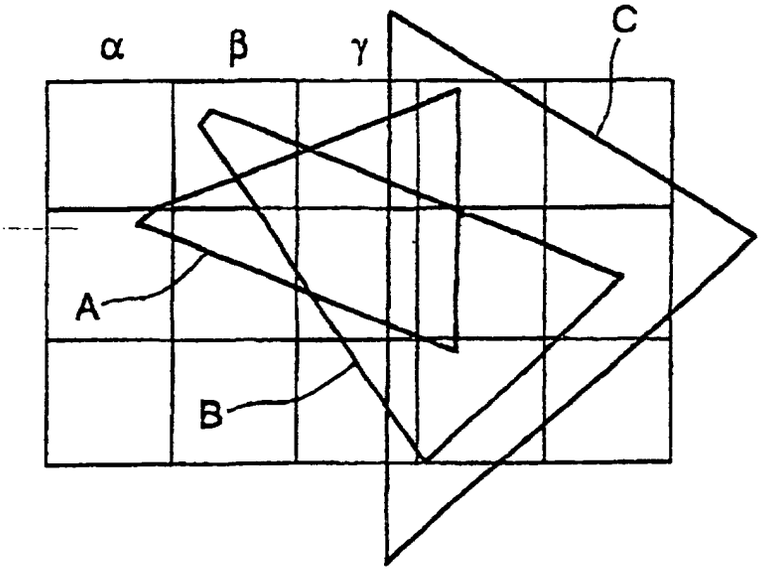



FIG.15

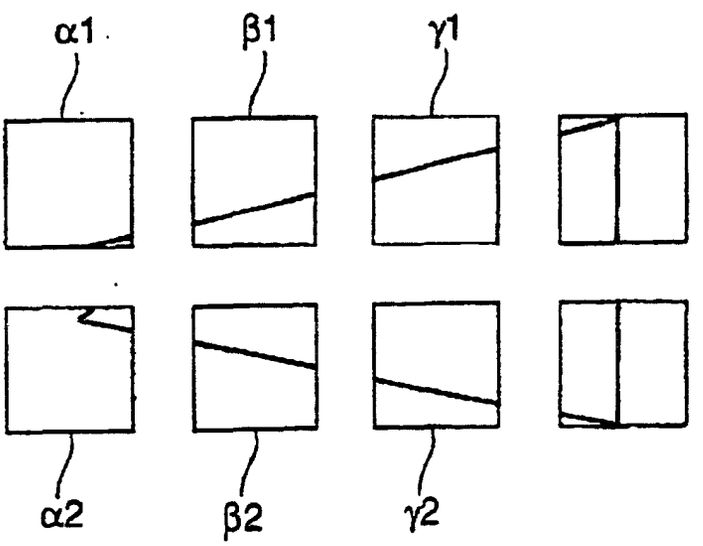


FIG.16

| MEMORY AREA FOR α |
| --- |
| MEMORY AREA FOR β |
| MEMORY AREA FOR γ |
| |
| |
| |

FIG.17