(12) PATENT
(19) AUSTRALIAN PATENT OFFICE

(11) Application No. AU 200116967 B2
(10) Patent No. 776830

(54) Title
Method for the real-time rendering of large volume data sets

(51)6 International Patent Classification(s)
G06T 017/00 G06T 011/00

(21) Application No: 200116967 (22) Application Date: 2000.10.18

(87) WIPO No: WO01/29773

(30) Priority Data

(31) Number	(32) Date	(33) Country
19950013	1999.10.18	DE

(43) Publication Date : 2001.04.30
(43) Publication Journal Date : 2001.07.19
(44) Accepted Journal Date : 2004.09.23

(71) Applicant(s)
Fraunhofer-Gesellschaft Zur Forderung Der Angewandten Forschung E.V.

(72) Inventor(s)
Bernd Froehlich; Michael Tirtasana

(74) Agent/Attorney
Davies Collison Cave,Level 15,1 Nicholson Street,MELBOURNE VIC 3000

(56) Related Art
US 5123084
US 4862392

AU 200116967

(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
26. April 2001 (26.04.2001)

PCT

(10) Internationale Veröffentlichungsnummer
WO 01/29773 A1

(51) Internationale Patentklassifikation[7]: G06T 17/00, 11/00

(21) Internationales Aktenzeichen: PCT/EP00/10254

(22) Internationales Anmeldedatum: 18. Oktober 2000 (18.10.2000)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
199 50 013.4 18. Oktober 1999 (18.10.1999) DE

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: ~~GMD — FORSCHUNGSZENTRUM INFORMATIONSTECHNIK GMBH [DE/DE]; Schloss Birlinghoven, 53754 Sankt Augustin (DE).~~

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **FROEHLICH, Bernd** [DE/DE]; Johannesstrasse 18, 53225 Bonn (DE). TIRA-SANA, Mildred-Scheel-Blumenweg 20 51067 Köln, Germany

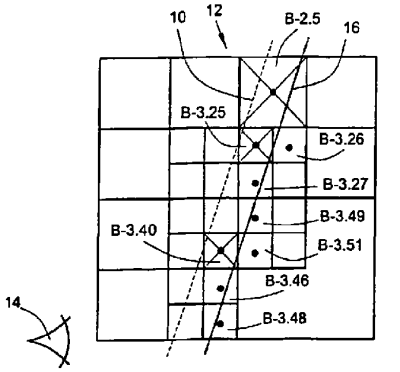
(74) Anwälte: **HILLERINGMANN, Jochen** usw.; Bahnhofsvorplatz 1 (Deichmannhaus), 50667 Köln (DE).

(81) Bestimmungsstaaten *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN,

[Fortsetzung auf der nächsten Seite]

(54) Title: METHOD FOR THE REAL-TIME RENDERING OF LARGE VOLUME DATA SETS

(54) Bezeichnung: VERFAHREN ZUR ECHTZEIT-DARSTELLUNG VON GROSSEN VOLUMENDATENMENGEN



A benötigte Blöcke	davon im B Zwischenspeicher	C zu laden
9	3	6

A... BLOCKS REQUIRED
B... BLOCKS THEREOF IN THE INTERMEDIATE MEMORY
C... TO BE LOADED

(57) **Abstract:** The invention relates to a method for the real-time rendering of large volume data sets. According to said method, a subset of volume data is selected in order to generate said rendering by means of said subset. The aim of the invention is to realize a maximum rendering repetition at a limited volume data charge rate and storage rate. To this end, volume data of different resolutions are used for the rendering. The low-resolution volume data mainly refer to areas of rendering that are less interesting for the viewer. For example, said areas of rendering are further away from the viewer than other areas which in turn are represented by higher resolution volume data.

(57) **Zusammenfassung:** Bei dem Verfahren zur Echtzeit-Darstellung grosser Mengen von Volumendaten wird eine Teilmenge von Volumendaten ausgewählt, um anhand dieser Teilmenge die Darstellung zu erzeugen. Um bei begrenzter Volumendaten-Laderate und -Speicherrate eine maximale Darstellungsfolgefrequenz zu realisieren, werden für die Darstellung Volumendaten unterschiedlicher Auflösung verwendet. Die Volumendaten mit geringerer Auflösung betreffen dabei hauptsächlich Bereiche der Darstellung, die für den Betrachter weniger interessant sind. Beispielsweise sind diese Bereiche der Darstellung weiter entfernt vom Betrachter als andere Bereiche, die dann entsprechend anhand von Volumendaten höherer Auflösung dargestellt werden.

(71) FRAUNHOFER-GESELLSCHAFT ZER FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V.
Leonrodstrasse 54, 80636 Munchen, Germany

WO 01/29773 A1

IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Bestimmungsstaaten** *(regional)*: ARIPO-Patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), eurasisches Patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI-Patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Veröffentlicht:

— *Mit internationalem Recherchenbericht.*

— *Vor Ablauf der für Änderungen der Ansprüche geltenden Frist, Veröffentlichung wird wiederholt, falls Änderungen eintreffen.*

Zur Erklärung der Zweibuchstaben-Codes, und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

ABSTRACT

Method for the real-time rendering of large volume data amounts

The invention relates to a method for the real-time rendering of large volume data amounts. According to said method, a partial amount of volume data is selected in order to generate said rendering by means of said partial amount. The aim of the invention is to realize a maximum rendering repetition at a limited volume data loading rate and storage rate. To this end, volume data of different resolutions are used for the rendering. The low-resolution volume data mainly refer to areas of rendering that are less interesting for the viewer. For example, said areas of rendering are further away from the viewer than other areas which in turn are represented by higher resolution volume data.

Method for the real-time rendering of large volume data amounts

Image rendering methods like computer tomography (CT) in the field of medicine, reflection measurements in the field of the petrol industry or computer simulations of flow simulations, for example, today provide very highly resolved three-dimensional data in an order of several hundred megabytes up to terabytes. The amounts of data provided are generally structured as a three-dimensional matrix having the measured values as its entries. Each value represents a scan value in a small space. These entries into the matrix are also referred to as voxels (volume picture data).

In general, the data are visualized by optional sections through the three-dimensional volume or by the so-called volume rendering which allows a semi-transparent three-dimensional rendering of the data.

Of particular interest is the real-time rendering of these data, where a plurality of section or volume rendering images are generated per second. Today, the real-time rendering is most often assisted by graphics hardware. However, present graphic boards have limited storage for three-dimensional volume data which, in this context, are also referred to as 3D textures. Typically, the storage size for 3D textures ranges from 1 MB in the low end section to 64 MB in high end graphics computers. At a resolution of 8 bits per voxel, one is reduced to a maximum resolution of 512x512x256 voxels for a 64 MB texture storage. Larger volumes cannot be rendered therewith or they can be rendered only with very low picture frequency.

It is desired to provide at least a useful alternative, and preferably a method that guarantees a high picture frequency even with volume data larger than the texture storage.

According to the present invention, there is provided a method for real-time rendering of large volume data sets, wherein

a) a partial amount of the volume data, by which the volume data are rendered within a rendering generation interval, is stored in a memory, the maximum volume data load rate at which volume data can be loaded into the memory within a load interval being as limited as the maximum amount of volume data storable in the memory, and a set of volume data can be defined within the partial amount of volume data, which are less relevant for image generation in a rendering generation interval than other volume data of the partial amount,

b) the volume data amount is hierarchically divided into blocks of volume data of different divisional steps, each block of one divisional level including the volume data of the blocks of the next lower divisional level to which the relevant block is associated, said volume data having a lower resolution that the volume data of the blocks of those next lower divisional level,

c) starting from the top divisional level, it is determined which blocks contain volume data required for the rendering in the next rendering generation interval, and the amount of volume data required for rendering in the next rendering generation interval is determined from the required blocks,

d) the amount of volume data required is compared to the maximum amount of volume data storable in the memory,

e) if the amount of required volume data is equal to the maximum amount of storable volume data, the process goes on to step h),

f) if the amount of required volume data is smaller than the maximum amount of storable volume data

- it is determined from the blocks of the next lower divisional level which blocks of this divisional level contain volume data required for rendering in the next rendering generation interval,

or

- should step g) have been executed already, the process continues at step h),

g) if the amount of required volume data is larger than the maximum amount of storable volume data, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step d),

h) the amount of volume data required for the rendering in the next rendering generation interval and to be loaded into the memory is determined from the blocks required for the rendering in the next rendering generation interval and not included in the memory,

i) the amount of volume data to be loaded is compared to the maximum volume data load rate,

j) if the amount of volume data to be loaded is smaller than or equal to the volume data load rate, the volume data required for the rendering in the next rendering generation interval and still to be loaded are loaded into the memory, and the rendering is effected in the next rendering generation interval,

k) if the amount of volume data is larger than the maximum volume data load rate, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step h).

A variant of the present invention provides a method for real-time rendering of large amounts of volume data, wherein

a) a partial amount of the volume data, by which a picture is generated within a rendering generation interval, is stored in a memory, the maximum volume data load rate at which volume data can be loaded into the memory within a load interval being as limited as the maximum amount of volume data storable in the memory, and a set of volume data can be defined within the partial amount of volume data, which are less relevant for image generation in a rendering generation interval than other volume data of the partial amount,

b) the volume data amount is hierarchically divided into blocks of volume data of different divisional steps, each block of one divisional level including the volume data of the blocks of the next lower divisional level to which the relevant block is associated, said volume data having a lower resolution that the volume data of the blocks of those next lower divisional level,

c) starting from the top divisional level, it is determined which blocks contain volume data required for the rendering in the next rendering generation interval,

d) the number of required blocks including volume data required for the rendering in the next rendering generation interval is compared to the

maximum number of storable blocks determined by the maximum number of volume data storable in the memory,

e) if the number of required blocks is equal to the maximum number of blocks, the process goes on to step h),

f) if the number of required blocks is smaller than the maximum number of blocks

– it is determined from the blocks of the next lower divisional level which blocks of this divisional level contain volume data required for rendering in the next rendering generation interval,

or

– should step g) have been executed already, the process continues at step h),

g) if the number of required blocks is larger than the maximum number of blocks, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step d),

h) the current block loading number of blocks required for the rendering in the next rendering generation interval and to be loaded into the memory is determined by determining which of the blocks required for the rendering in the next rendering generation interval are not included in the memory,

i) the current block loading number is compared to maximum block loading number corresponding to the maximum volume data load rate,

j) if the current block loading number is smaller than or equal to the maximum block loading number, the blocks required for the rendering in the next rendering generation interval and still to be loaded are loaded into the memory, and the rendering is effected in the next rendering generation interval,

k) if the current block loading number is larger than the maximum block loading number, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step h).

For the preferred embodiments described herein, it is assumed that the amount of volume data required for rendering is limited due to the capacity of a (memory) memory in which the volume data required for rendering are stored. Further, the rate at which volume data are loaded into the memory per loading interval is limited as well. To be able to guarantee high picture frequencies despite these marginal conditions, a preferred method suggests to render volume data that are less relevant to the rendering in a rendering generation interval in a coarser resolution than the volume data required for the rendering. For example, such portions of a rendering are less relevant that are farther away from the viewer. Anyway, the method always guarantees that the relevant volume data are rendered at a higher resolution, but with at least the same resolution than the less relevant volume data. To this end, the entire volume data set is first hierarchically subdivided into blocks of volume data of different divisional levels and, thus, different resolution levels. In the top divisional level, the resolution is lowest. In contrast therewith, the lowest divisional level guarantees the highest resolution

and contains the individual volume data themselves, for example. With a view to the execution of the method, it is feasible for each block of each divisional level to have the same number of volume data. In the following, this structure will also be referred to as "Octree".

First, it is determined in which divisional level there are fewer blocks containing volume data necessary for rendering than determined by the maximum number of volume data storable in the memory. Then, the next lower divisional level with the next higher resolution of volume data is considered. Thus, more volume data are required for rendering than are storable in the memory. The volume data less relevant for rendering will now be rendered coarser than the others. Thus, the amount of volume data is reduced successively until this amount is smaller than the capacity of the memory. When this is achieved, the required amount of volume data is examined so as to determine which of the required volume data have to be loaded into the memory. It is well possible that the memory is already loaded with volume data that were required for the rendering of volume data in the previous rendering generation interval and are also required for the rendering of volume data in the next rendering generation interval. If the differential amount of volume data is greater than loading rate, i.e. the amount of volume data that can be loaded per loading interval, the amount of volume data required for the rendering in the next rendering generation interval has to be reduced further according to the above pattern by making the resolution of less relevant volume data coarser. This process is continued until the still to be loaded amount of volume data required for rendering in the next rendering generation interval is at most equal to the loading rate.

A variant of the preferred method provides that the amount of volume data is determined by the number of blocks, wherein all blocks contain the same amount of volume data and, preferably in the top divisional level, a block is provided that includes the entire volume data set.

The volume data less relevant for rendering can be defined by the user. For example, these may be the highest-resolution volume data that are farthest away from the viewer. Alternatively, one may define certain portions of the volume data set as less relevant, regardless of their distance from the viewer. The volume data defined as less relevant can also differ from rendering interval to rendering interval. In particular, the criteria by which volume data are defined as less relevant for rendering may be the same or different within the individual rendering intervals.

The reduction of the amount of volume data required for rendering is "bought" with a coarsening of the resolution. Thus, it may well happen that in one rendering of the volume data, these are represented with different levels of resolution. It has proven feasible to merely allow renderings of the volume data that only include two different stages of resolution.

The value of the loading rate, i.e. the amount of volume data storable into the memory during a loading interval, must be chosen depending on how much the volume data amount has increased per block from divisional level to divisional level. If, for example, a block on the next lower divisional level is comprised of eight blocks, the loading rate should be such that the volume data of at least eight blocks can be loaded in one loading interval.

The amount of volume data storable in the memory should be sufficiently large so that, for example, a sectional plane through the volume data set can be rendered with a sufficiently high resolution. Regarding the size of the memory embodied as the memory on the graphics board of a PC, for example, comments have been made in the introduction to which reference is made therefor.

Preferred embodiments of the present invention are hereinafter described, again by way of example only, with reference to the accompanying drawings, wherein:

Fig. 1 illustrates the division of volume data into individual blocks for a three-dimensional case,

Fig. 2 illustrates the division of two-dimensional data into individual blocks,

Figs. 3 - 25 illustrate a first embodiment for building a level rendering without displacement, the level being displaced afterwards,

Figs. 26 - 48 illustrate a second embodiment for building a level rendering without displacement, the level being displaced afterwards,

Figs. 49 - 60 illustrate a third embodiment of the rendering of a level with displacement, and

Figs. 61 - 82 illustrate a fourth embodiment for building a level rendering without displacement.

Fig. 1 illustrates how, in a three-dimensional case, volume data in individual divisional levels (four divisional plans from −3 to 0, in the present case) are divided hierarchically in blocks (bricks). Each block of each divisional level contains the same number of volume data, these volume data differing from divisional lever to divisional level.

In the embodiments, it is assumed that 512 blocks exist on the lower divisional level and that these blocks B-3.1 to B-3.512 each contain the scanned values of a volume, i.e. the individual voxels. Eight blocks are comprised into one block of the next higher divisional level -2, respectively, where the number of volume data of one block of divisional level -2 is equal to the number of volume data of one block of the lowermost divisional level. In the divisional level -2, there exist 4 blocks (B-2.1 to B-2.64), where the block B-2.1 represents the volume data of blocks B-3.1 to B-3.8 with a higher resolution. Eight volume

data of a block of the lowermost level -3 are thus represented by a volume data bit of a block of the next higher divisional level -2.

According to the above scheme, eight blocks of the divisional level 1 are comprised together, respectively. Thus, block B-1.1, for example, consists of blocks B-2.1 to B-2.8. On the top divisional level 0, there exists only one block B0 formed by eight blocks B-1.1 to B-1.8. As a result, the volume data become ever coarser from a lower divisional level to a higher divisional level, i.e. they can be rendered with increasing resolution. Such nesting of blocks is also referred to as "Octree".

In other words, the texture memory (volume data set) is divided into an amount of blocks of equal size, the so-called bricks. Every brick can include an own three-dimensional texture of a certain size (e.g. 64x64x64). The volume data set (e.g. 1024x1024x1024 on the first divisional level) is preferably divided into blocks of the same size so that a respective one of these volume data set blocks fits into a brick. Further, 2x2x2 adjacent blocks of the volume data are again comprised into a block with the same resolution. Thereby, on the next lower divisional level, a data set is obtained having one eighth of the original resolution, where, again, 2x2x2 adjacent blocks are comprised, and so on. This results in a hierarchy that has been referred to as "Octree" above. Based on the example of a 1024x1024x1024 voxel data volume, 16x16x16 blocks are formed on the finest-resolution level, 8x8x8 blocks on the next level, then 4x4x4 blocks, then 2x2x2 blocks, and on the topmost level there is only one block. The resolution of the data set is thus reduced successively from 1024x1024x1024 to 512x512x512 to 256x256x256, 128x128x128, and 64x64x64 on the topmost level. Considering, for example, a sectional plane through the data set, only few blocks of the data volume will be sectioned in general. Only these will be loaded into the texture memory and used for rendering the data on the sectional plane. The so-called volume rendering is effected by superimposing a plurality of such semi-transparent sectional

planes and can thus be treated as a plurality of sectional planes, the method may also contemplate polygonal bodies instead of planes, on whose surface the volume data are to be rendered. For all sectional planes to be rendered, the blocks that are relevant for the rendering are determined. However, two important conditions must be observed:

1. The total number of blocks required has to fit into the texture memory.

2. Loading these blocks takes time so that from one picture to the next only a certain number of new blocks can be loaded into the texture memory, without allowing the picture frequency to drop below a certain limit.

Both problems are solved with the method described. To this avail, the sectional planes (or other polygonal objects) are sorted step-wise from the "top", i.e. from the topmost divisional level, into the octree until the maximum allowable number of blocks (condition 1) is reached. Here, blocks that are closer to the eye of the viewer are divided first. Then, it is determined which of the blocks have been loaded already in the last cycle. If more new blocks are required than allowable (condition 2), blocks are again comprised until the second condition is met. Comprising is done first for blocks that are farther away from the viewer. The division steps and the subsequent comprising can also be integrated in one procedure. Upon a fast movement of a sectional plane, the second condition causes generally "coarse", low-resolution blocks to be rendered, which correlates very well with the generally known movement blur. When the movement of the sectional plane is stopped, the blocks with better resolution are sequentially loaded into the texture memory until the first condition is fulfilled.

For the two-dimensional case, for which some embodiments of the method will be explained hereafter, a situation as illustrated in Fig. 2 is obtained. Here,

individual blocks that correspond to a block in the 3D case bear the same reference numerals. The denomination of the blocks on the individual divisional levels are also evident from Fig. 2; reference will be made to these denominations hereafter.

In the two-dimensional case, for example, a sectional plane through a volume data set appears as a line. In the following Figures, this line is to represent a sectional plane through a volume data set.

In the first embodiment of Figs. 3 to 25 described hereinbelow, a sectional plane 10 represented by a line and extending through a volume data set referenced as 12 is to be rendered, the viewer being indicated at 14. It is assumed that a maximum of nine blocks can be stored in the memory that holds the volume data required for rendering the plane 10. A further parameter is that a maximum of 5 blocks of volume data can be loaded in one loading interval. The volume data less relevant for the rendering of plane 10 are assumed to be those volume data that are farther away from the viewer 14.

In the present method according to the first embodiment, the amount of volume data is determined first that is necessary for rendering the plane 10, if one assumes that this is based on the volume data included in the (only) block B0 of the topmost divisional level. Thus, the rendering requires one block. Under the precondition that the memory is still empty, one may state that none of the blocks required for rendering the plane 10 is stored in the memory. As a result, one block has to be loaded into the memory to subsequently render the plane 10. This fact, together with the respective required blocks, those blocks of the required blocks that are stored in the memory and those blocks still to be loaded, is indicated in a table in Fig. 3 and in the following Figures 4 to 25.

Since it is the aim of the method to be able to render the plane 10 with as high a resolution as possible, one tries to store as many pictures with high resolution into the memory as possible, where those volume data required for rendering the plane 10 that are close to the viewer 14 should have a higher resolution than the volume data farther away from the viewer 14. Under the above conditions and defaults, it is thus evident that rendering the plane 10 only through block B0 of the top divisional level is insufficient and that the condition to be able to load a maximum of 9 blocks into the memory is not met by far. Therefore, based on the step of Fig. 3, the block B0 is divided into four blocks B-1.1 to B-1.4 of the next lower divisional level -1.

This results in the situation of Fig. 4, from which it is obvious that now the rendering of the plane 10 requires he volume data of three of the blocks of the divisional level -1. Of these blocks, none is stored in the memory so that three blocks have to be loaded. Since also in this phase of the method, the maximum allowable number of blocks has not been reached, the first three blocks of the divisional level -1 are further divided.

This results in the situation of Fig. 5. The required blocks are marked by a point in the drawing, just as in the Figs. 3 and 4 and the other Figs.. Now, according to Fig. 5, five blocks of the divisional level -2 are required for rendering the plane 10. Again it is true that none of these blocks is yet stored in the memory, which is why five blocks have to be loaded.

Since the maximum number of storable blocks is still not reached, a further division is effected, as illustrated in Fig. 6, namely into blocks of the next lower divisional level -3, which in this embodiment is the lowermost divisional level. Now, it is obvious that rendering the plane 10 based on the volume data of the blocks of the lowest divisional plane requires ten such blocks. Thus, the parameter that a maximum of nine blocks can be stored in the memory is

exceeded. It must now be tried to reduce the number of blocks required for rendering the plane 10.

According to Fig. 7, this is realized by replacing the blocks B-3.17 and B-3.19 that are farthest away from the viewer 14 with that block of the next higher divisional level whose volume data also include the volume data of blocks B.-3.17 and B-3.19, yet with a coarser resolution. In other words: the volume data of blocks B-3.17 and B-3.19 are replaced with the volume data of block B-2.5. the remaining blocks remain unchanged so that the total number of blocks, i.e. the amount of volume data necessary for rendering plane 10, i reduced by one block. Thus, the condition that a maximum of nine blocks can be stored in the memory is met. However, since none of these nine blocks is stored in the memory, nine blocks would have to be loaded in one loading interval. This is contradictory to the second condition, according to which a maximum of 5 blocks can be loaded in one loading interval.

Thus, the number of blocks required for rendering is further reduced, as illustrated in Fig. 8. In this case, the volume data of block B-3.25 have been replaced with the volume data of the associated block B-2.7 of the next higher divisional level. Unfortunately, this replacement results in no reduction of the presently required number of blocks and thus it results in no reduction of volume data as is evident from the table in Fig. 8 and Fig. 8 itself. Still, nine blocks are required. This requires the loading of nine blocks in one loading interval, which is not allowed.

For this reason, the process is carried on as above and as illustrated in Fig. 9. Now, the volume data of blocks B-3.14 and B-3.16 have been replaced with the volume data of the associated block B-2.4 of the next higher divisional level. The volume data of blocks B-3.14 and B-3.16 are the blocks of the divisional level -3 that are farthest away from the viewer 14. Since two blocks

of the divisional level -3 are replaced with a block of the divisional level -2, the total amount of volume data is reduced by the volume data of one block.

In Fig. 9, one could basically have replaced the blocks B-2.5 and B-2.7 of the divisional level -2 with the block B-1.2 of the divisional level -1, which comprises these blocks. However, one would have needed blocks for rendering the plane 10 whose volume data would differ by two resolution steps. This may have undesired effects when rendering the plane 10, which is why in this embodiment (and in other embodiments) it is assumed that the blocks required for rendering a plane comprise volume data with a total of 2 adjacent resolution steps at most.

As is evident from Fig. 9, there are still too many blocks to be loaded, namely eight. Thus, in the next step (see Fig. 10), the block B-2.10 of the next higher divisional level, which includes the three blocks B-3.38 to B-3.40, is used instead the same. Thus, the total number of required blocks is reduced by 3 (with reference to the step of Fig. 9), but there are still six blocks to be loaded.

Therefore, in the next step (see Fig. 11), the block B-2.12 is introduced that replaces the two blocks B-3.45 and B-3.47. Now, the number of blocks required can be reduced to five, none of which is yet stored in the memory. In other words, five blocks have to be loaded, thereby reaching the exact number of blocks that can be loaded in one loading interval. The blocks listed in the table of Fig. 11 will now be loaded into the memory and the plane 10 is rendered using these blocks.

As is evident from a comparison of Figs. 11 and 5, the situation according to Fig. 11 was already existent in the step of Fig. 5. To shorten the process, one could have decided directly from the situation of Fig. 5 to load the five required blocks into the memory. This is surely feasibly in some instances and it is advantageous with respect to a fast building of the blocks required

for rendering the plane 10. However, it is feasible to first try use up the maximum number of blocks storable in the memory for rendering the plane, and to examine only afterwards how many of these blocks must still be loaded. In this manner, it is guaranteed that the volume data relevant for rendering are present with a higher resolution than the less relevant data.

After five blocks have been loaded in the last step (Fig. 11) and the plane 10 has been rendered based on the volume data of those five blocks, all blocks of the second divisional level are divided in blocks of the next smaller divisional level in the next step illustrated in Fig. 12. Then, it is again checked how many blocks are now required for rendering. In the present case, again, ten blocks of the divisional level -3 are required. Since ten blocks are more than the number of blocks storable in the memory at the same time, a reduction of the division has to be started again, as illustrated in Fig. 13, pertaining to volume data farthest from the viewer 14. In other words, the two blocks B-3.17 und B-3.19 are replaced with the block B-2.5 of the divisional level 2, whereby the total number of blocks required is reduced by one. Thus, the first condition is met. Of the nine blocks required now, one, namely block B-2.5, is already in the memory (in Fig. 13, this block is marked with an X; for Fig. 13 and all other Figs., it is true that blocks already in loaded into the memory and required for rendering a plane are marked by crossing as already present in the memory) so that a total of eight blocks has to be loaded. This is in conflict with the second condition according to which only five blocks can be loaded in one loading interval.

Therefore, in the next step (Fig. 14), the block B-3.25 of the divisional level -3 that is farthest from the viewer 14 is transferred into the associated block B-2.7 of the divisional level -2. Besides the fact that this results in no reduction of the blocks required for rendering the plane 10, it is achieved that two of the required blocks, i.e. the blocks B-2.5 and B-2.7 are already stored in the memory. Thus, seven blocks have to be loaded still. These are still too many,

which is why in the next step in Fig. 15, the two blocks B-3.14 and B-3.16 of the divisional level -3, which are farthest away from the viewer 14, are transferred into the associated block B-2.4 of the divisional level -2. Thus, three of the eight required blocks are stored in the memory, so that five remain to be loaded. Now, the loading condition is met and blocks B-3.38 to B-3.40, B-3.45 and B-3.47 are loaded into the memory.

Starting from the situation of Fig. 11, one may also arrive at the situation of Fig. 15 by first dividing the block B-2.12 closest to the viewer 14 into the two associated blocks of the next divisional level. Rendering would then take seven blocks which are fewer than the maximum number of blocks storable in the memory at the same time. Accordingly, in a next step, the block B-2.10 of the divisional level -2, which is now closest to the viewer 14, into the associated three blocks of the divisional level -3. Thus, the number of blocks required for rendering the plane 10 increases to 3 and the situation of Fig. 15 is achieved.

In the next step, illustrated in Fig. 16, the three blocks of the divisional level -2, which in Fig. 15 are still existent, are divided into the corresponding blocks of the divisional level -3. As a result, ten blocks are again required for rendering the plane 10. Of these blocks, those marked with an X are already stored in the memory so that five blocks remain to be loaded. At this moment, this is of no importance, since ten blocks are required for rendering, i.e. more than the allowable maximum number of blocks.

In Fig. 17, the two blocks of the divisional level -3, which are farthest from the viewer 14, are transferred into the associated block of the divisional level -2. Of the nine blocks required, the blocks listed in the table and symbolized in Fig. 17, are already stored in the memory so that a total of three blocks have to be loaded still. This is possible in one loading interval so that blocks B-3.14, B-3.16 and B-3.25 are loaded now.

In step 18, it is again checked whether the portion farthest away from the viewer 14 can be rendered with a higher resolution in the rendering of the plane. To this end, block B-2.5 is replaced with the corresponding blocks of the next lower divisional level. Of these four blocks, two are needed for rendering so that, again, a total of ten blocks is needed, which are too many. Thus, in step 19, the above division is cancelled. Of the nine blocks required, all nine are stored in the memory so that none has to be loaded.

The previous explanation of the present method according to this embodiment is based on the fact that the plane to be rendered does not shift. Thus, this example illustrates how a plane that initially has a rather low resolution when rendered, can be rendered with an ever higher resolution. Should the plane to be viewed shift, the same measures as descried are eventually required. The shifting of the plane requires that the rendering of the shifted plane (new plane) can be based to a much lesser extent on the volume data that were already required for rendering the previous plane (old plane). Figs. 20 to 25 illustrate an example of a plane shifting. Plane 16 represents the new plane, whereas in Fig. 20, for example, the old plane 10 is plotted as well. To render the plane 16, the blocks marked by an "X" and listed in the table in Fig. 20 are required. Here, nine blocks are needed, three of which are already stored in the memory. Accordingly, six have to be loaded still. Since these are too many, those blocks of the divisional level -3 that are farthest from the viewer 14 are replaced with the associated block of the divisional level -2, following the well-known pattern. Thus, the number of volume data required for rendering is reduced to seven blocks, two of which are already stored in the memory. The remaining five blocks can be loaded simultaneously in one loading interval. Thereafter, the plane 16 is rendered.

Assuming that the plane 16 is not moved further, the rendering thereof can be made with a higher resolution. This is again effected by examining, as illustrated in Fig. 22, which of the blocks of the next lower divisional level

are required for rendering the plane 16. These are a total of ten blocks, requiring more space than existing in the memory. Accordingly, in the step of Fig. 23, in the portion of the volume data farthest from the viewer 14, the two blocks B-3.18 and B-3.20 are replaced with block B-2.5. Thus, nine blocks are required for rendering the plane 16, six of which are already stored in the memory so that three have to be loaded still. Thereafter, according to Fig. 24, it is tried to increase the resolution of the rendering in the portion farthest form the viewer 14. However, this fails eventually because this rendering requires more blocks, namely ten, than can be accommodated by the memory. The division in to the blocks of the divisional level -3 effected in Fig. 24 is thus cancelled. This concludes the building of the rendering of plane 16.

Figs. 26 to 48 illustrate a second embodiment of the building of the rendering of a plane in a volume data set as well as the conditions upon rotating this plane. The tables listed in those Figs. only indicate the number of blocks respectively required for the rendering, the number of blocks already stored in the memory and the number of blocks to be loaded. Again, it is assumed that the memory is empty initially. It is a precondition that a maximum of nine blocks can be stored in the memory. Different from the previous embodiment, the loading rate is restricted to four blocks. Blocks already stored and necessary for the rendering of the plane in the individual steps of the method are again crossed, i.e. marked with an "X". As far as applied, the reference numeral in Figs. 26 to 48 are identical with the reference numerals in Figs. 3 to 25.

An alternative to rotating the plane according to Figs. 45 to 48 is illustrated in Figs. 49 to 60. What has been stated above referring to the representation and use of reference numerals also applies to these Figs.. In the embodiment of Figs. 49 to 60, the plane 10 is shifted in parallel by a rather short distance. The parameters are that, again, a maximum of nine blocks can be stored in the memory at the same time and that a maximum of four blocks can be

loaded in one loading interval. After the shifted plane 16 has been represented for the first time (see the situation of Fig. 15), it is assumed, as in the embodiments of Figs. 26 to 48 that the plane is not moved further, i.e. that it may be built in rendering from a coarse resolution to a rather fine resolution. The blocks required for rendering the shifted plane 16 with the maximum resolution are visible in Fig. 60.

Figs. 61 to 82 illustrate a fourth application of the present method. Again, the same reference numerals as in the other Figs. have been used. Again, nine blocks may be stored in the memory at the same time, while only four blocks can be loaded any one time. It is assumed that the plane 10 to be rendered is not displaced. The blocks required for rendering the plane with the maximum resolution have been marked in Fig. 82.

The reference to any prior art in this specification is not, and should not be taken as, an acknowledgment or any form of suggestion that that prior art forms part of the common general knowledge in Australia.

Claims

1. Method for real-time rendering of large volume data sets, wherein

   a) a partial amount of the volume data, by which the volume data are rendered within a rendering generation interval, is stored in a memory, the maximum volume data load rate at which volume data can be loaded into the memory within a load interval being as limited as the maximum amount of volume data storable in the memory, and a set of volume data can be defined within the partial amount of volume data, which are less relevant for image generation in a rendering generation interval than other volume data of the partial amount,

   b) the volume data amount is hierarchically divided into blocks of volume data of different divisional steps, each block of one divisional level including the volume data of the blocks of the next lower divisional level to which the relevant block is associated, said volume data having a lower resolution that the volume data of the blocks of those next lower divisional level,

   c) starting from the top divisional level, it is determined which blocks contain volume data required for the rendering in the next rendering generation interval, and the amount of volume data required for rendering in the next rendering generation interval is determined from the required blocks,

   d) the amount of volume data required is compared to the maximum amount of volume data storable in the memory,

e) if the amount of required volume data is equal to the maximum amount of storable volume data, the process goes on to step h),

f) if the amount of required volume data is smaller than the maximum amount of storable volume data

- it is determined from the blocks of the next lower divisional level which blocks of this divisional level contain volume data required for rendering in the next rendering generation interval,

or

– should step g) have been executed already, the process continues at step h),

g) if the amount of required volume data is larger than the maximum amount of storable volume data, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step d),

h) the amount of volume data required for the rendering in the next rendering generation interval and to be loaded into the memory is determined from the blocks required for the rendering in the next rendering generation interval and not included in the memory,

i) the amount of volume data to be loaded is compared to the maximum volume data load rate,

j) if the amount of volume data to be loaded is smaller than or equal to the volume data load rate, the volume data required for the rendering in the next rendering generation interval and still to be loaded are loaded into the memory, and the rendering is effected in the next rendering generation interval,

k) if the amount of volume data is larger than the maximum volume data load rate, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step h).

2. Method for real-time rendering of large amounts of volume data, wherein

a) a partial amount of the volume data, by which a picture is generated within a rendering generation interval, is stored in a memory, the maximum volume data load rate at which volume data can be loaded into the memory within a load interval being as limited as the maximum amount of volume data storable in the memory, and a set of volume data can be defined within the partial amount of volume data, which are less relevant for image generation in a rendering generation interval than other volume data of the partial amount,

b) the volume data amount is hierarchically divided into blocks of volume data of different divisional levels, each block of one divisional level including the volume data of the blocks of the next lower divisional level to which the relevant block is associated, said volume data having a lower resolution that the volume data of the blocks of those next lower divisional level,

c) starting from the top divisional level, it is determined which blocks contain volume data required for the rendering in the next rendering generation interval,

d) the number of required blocks including volume data required for the rendering in the next rendering generation interval is compared to the maximum number of storable blocks determined by the maximum number of volume data storable in the memory,

e) if the number of required blocks is equal to the maximum number of blocks, the process goes on to step h),

f) if the number of required blocks is smaller than the maximum number of blocks

- it is determined from the blocks of the next lower divisional level which blocks of this divisional level contain volume data required for rendering in the next rendering generation interval,

or

- should step g) have been executed already, the process continues at step h),

g) if the number of required blocks is larger than the maximum number of blocks, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step d),

h) the current block loading number of blocks required for the rendering in the next rendering generation interval and to be loaded into the memory is determined by determining which of the blocks required for the rendering in the next rendering generation interval are not included in the memory,

i) the current block loading number is compared to maximum block loading number corresponding to the maximum volume data load rate,

j) if the current block loading number is smaller than or equal to the maximum block loading number, the blocks required for the rendering in the next rendering generation interval and still to be loaded are loaded into the memory, and the rendering is effected in the next rendering generation interval,

k) if the current block loading number is larger than the maximum block loading number, one determines, instead of those blocks that include the set of volume data less relevant to the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block, and thereafter continues with step h).

3. The method of claim 1 or 2, characterized in that, in step h), if the blocks required for rendering in the next rendering generation interval already contain blocks of different divisional levels, one determines, instead of those blocks of the lowest divisional level that include the set of volume data less relevant for the rendering in the next rendering generation interval, the next higher divisional level block associated with these blocks as the required block.

4. The method of one of claims 1 to 3, characterized in that each block of each divisional level comprises the same amount of volume data.

5. The method of one of claims 1 to 4, characterized in that one block exists in the topmost divisional level.

6. The method of one of claims 1 to 5, characterized in that it is true for each next lower divisional level that an equal number of blocks is associated with a block of the next higher divisional level.

7. The method of one of claims 1 to 6, characterized in that such volume data that are farthest from a viewer are defined as a group of volume data less relevant for rendering in the next rendering generation interval.

8. The method of one of claims 1 to 7, characterized in that, in the rendering generation intervals, different volume data are defined as a group of volume data less relevant for rendering in the next rendering generation interval.

9. Method for real-time rendering substantially as hereinbefore described with reference to the accompanying drawings.

DATED this 28th day of July 2004

FRAUNHOFER-GESELLSCHAFT ZUR FORDERUNG
DER ANGEWANDTEN FORSCHUNG E.V.
By its Patent Attorneys
DAVIES COLLISON CAVE

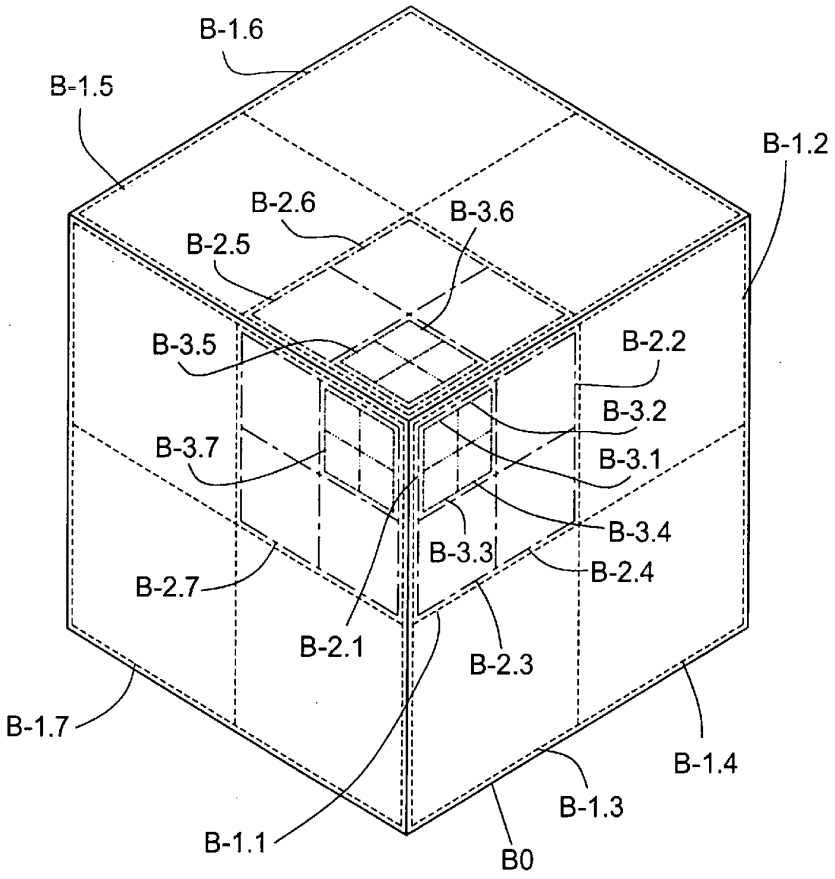

B-1.6
B-1.5
B-1.2
B-2.6
B-3.6
B-2.5
B-3.5
B-2.2
B-3.2
B-3.7
B-3.1
B-3.4
B-3.3
B-2.4
B-2.7
B-2.1
B-2.3
B-1.7
B-1.4
B-1.3
B-1.1
B0

FIG.1

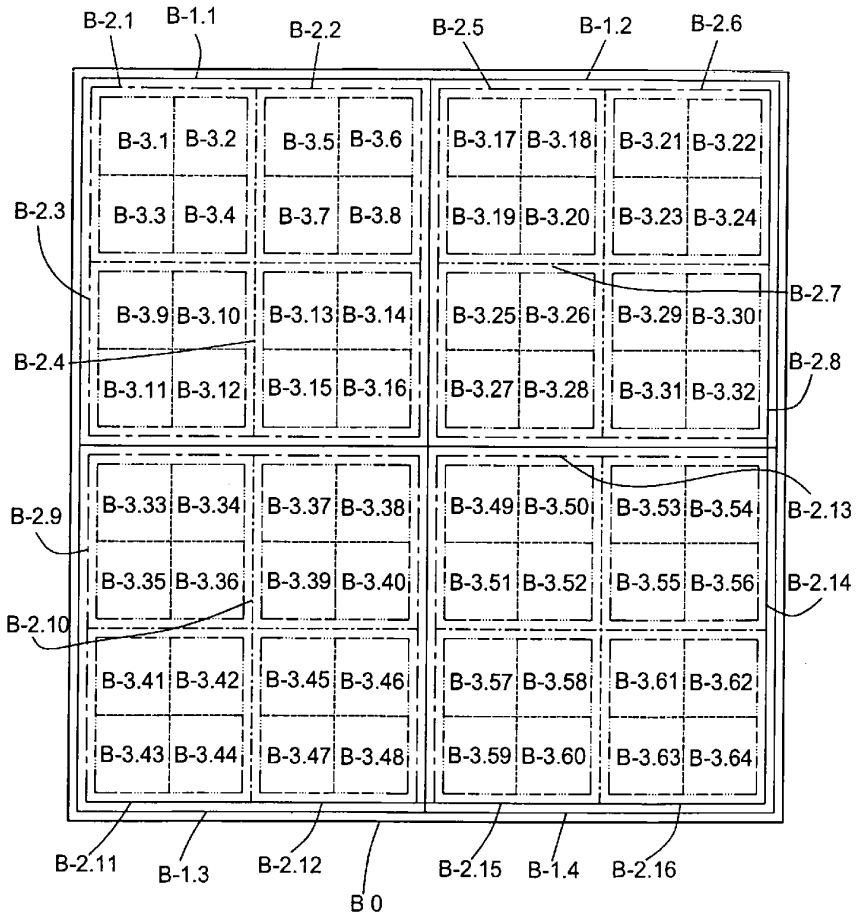
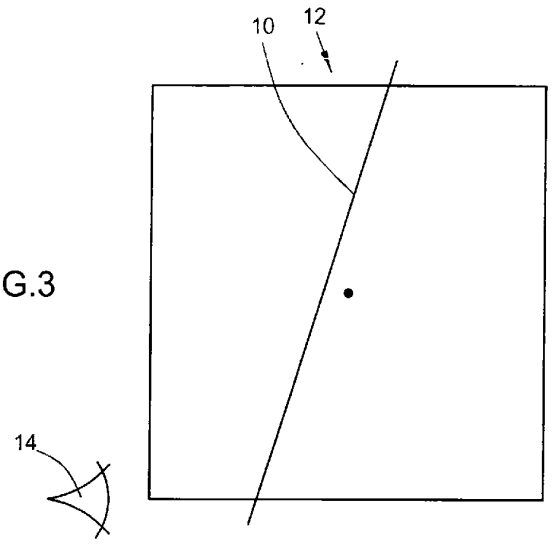

B-2.1
B-1.1
B-2.2
B-2.5
B-1.2
B-2.6
B-3.1 B-3.2 B-3.5 B-3.6 B-3.17 B-3.18 B-3.21 B-3.22
B-3.3 B-3.4 B-3.7 B-3.8 B-3.19 B-3.20 B-3.23 B-3.24
B-2.3
B-2.7
B-3.9 B-3.10 B-3.13 B-3.14 B-3.25 B-3.26 B-3.29 B-3.30
B-2.4
B-2.8
B-3.11 B-3.12 B-3.15 B-3.16 B-3.27 B-3.28 B-3.31 B-3.32
B-3.33 B-3.34 B-3.37 B-3.38 B-3.49 B-3.50 B-3.53 B-3.54
B-2.9
B-2.13
B-3.35 B-3.36 B-3.39 B-3.40 B-3.51 B-3.52 B-3.55 B-3.56
B-2.14
B-2.10
B-3.41 B-3.42 B-3.45 B-3.46 B-3.57 B-3.58 B-3.61 B-3.62
B-3.43 B-3.44 B-3.47 B-3.48 B-3.59 B-3.60 B-3.63 B-3.64
B-2.11
B-1.3
B-2.12
B-2.15
B-1.4
B-2.16
B 0

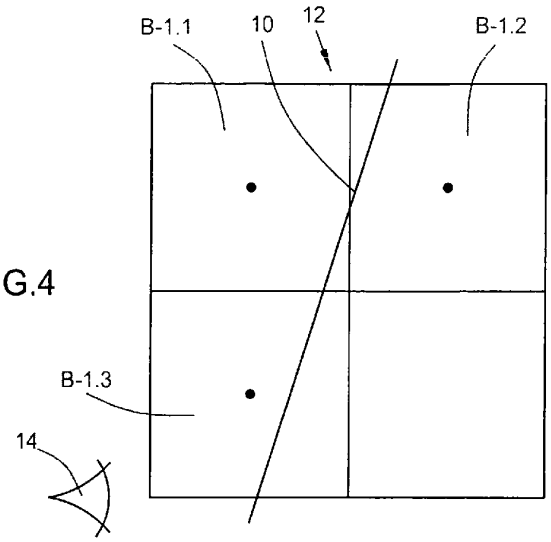
FIG.2

FIG.3



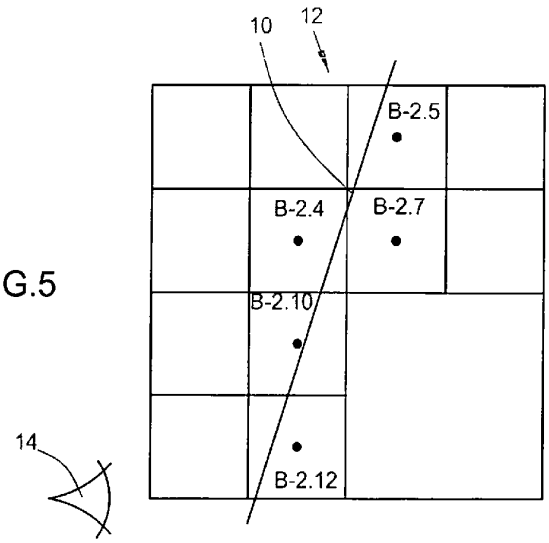
Blocks required	in the memory	to be loaded
B0	—	B0
1	0	1

FIG.4



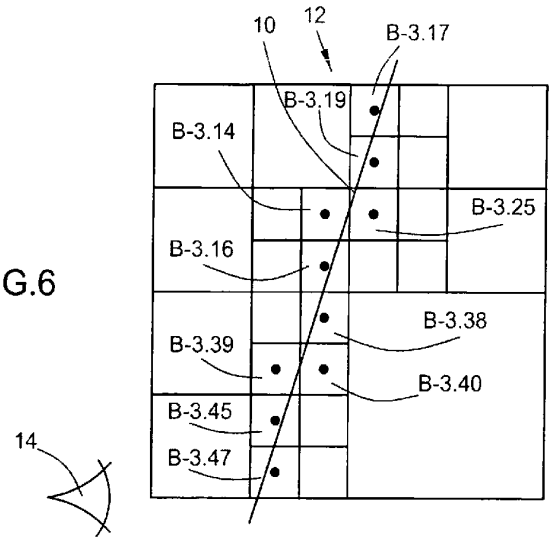
Blocks required	in the memory	to be loaded
B-1.1	——	B-1.1
B-1.2		B-1.2
B-1.3		B-1.3
3	0	3

FIG.5



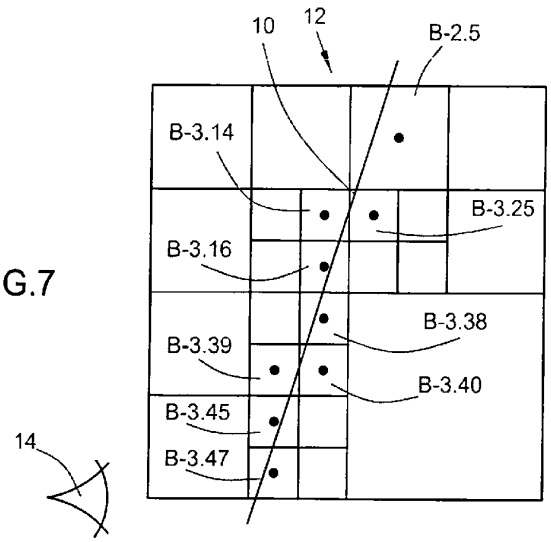
Blocks required	in the memory	to be loaded
B-2.4		B-2.4
B-2.5		B-2.5
B-2.7	—	B-2.7
B-2.10		B-2.10
B-2.12		B-2.12
5	0	5

FIG.6



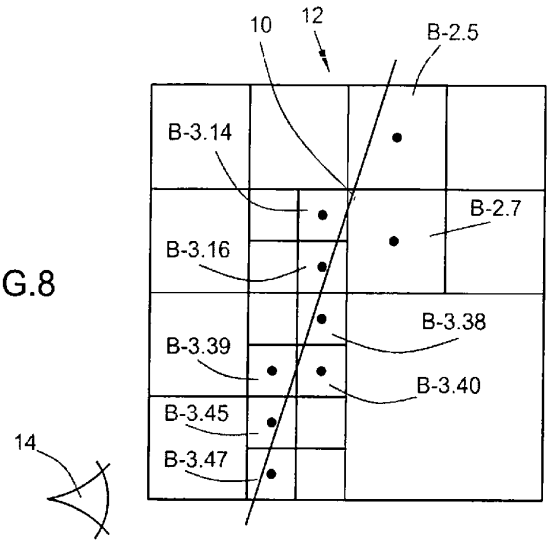
Blocks required	in the memory	to be loaded
B-3.14		B-3.14
B-3.16		B-3.16
B-3.17		B-3.17
B-3.19		B-3.19
B-3.25		B-3.25
B-3.38		B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
10	0	10

FIG.7



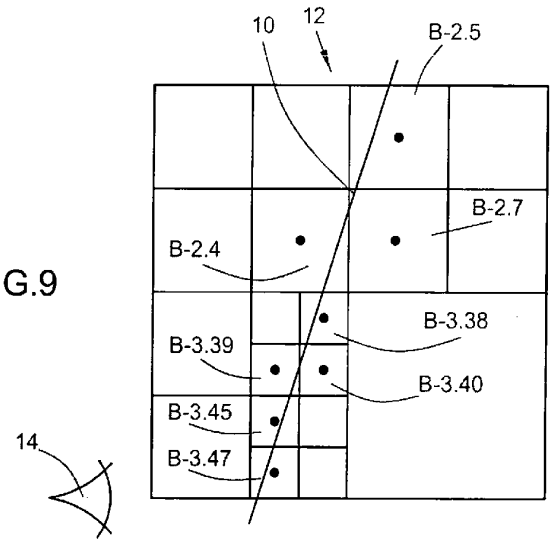
Blocks required	in the memory	to be loaded
B-2.5		B-2.5
B-3.14		B-3.14
B-3.16		B-3.16
B-3.25		B-3.25
B-3.38	—	B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
9	0	9

FIG.8



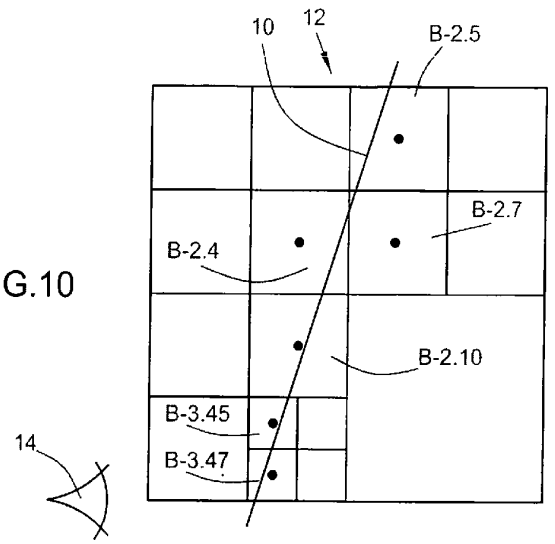
Blocks required	in the memory	to be loaded
B-2.5		B-2.5
B-2.7		B-2.7
B-3.14		B-3.14
B-3.16		B-3.16
B-3.38		B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
9	0	9

FIG.9



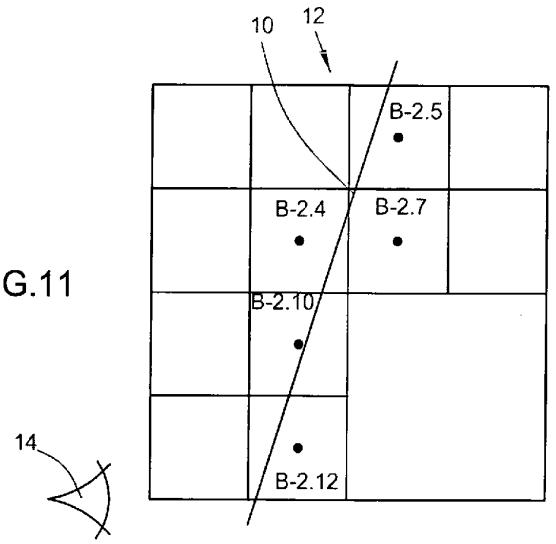
Blocks required	in the memory	to be loaded
B-2.4		B-2.4
B-2.5		B-2.5
B-2.7		B-2.7
B-3.38	—	B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
8	0	8

FIG.10



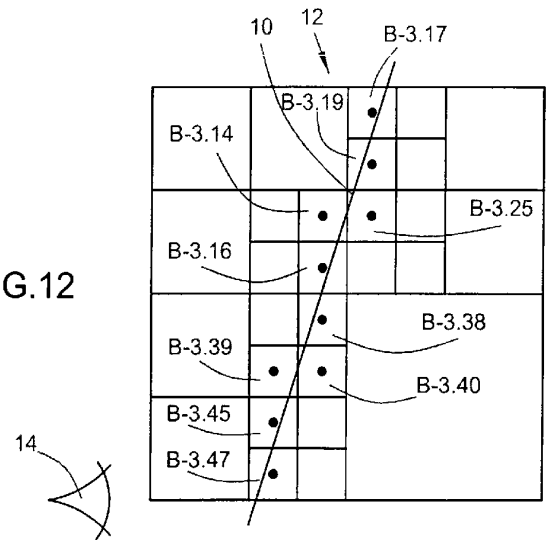
Blocks required	in the memory	to be loaded
B-2.4		B-2.4
B-2.5		B-2.5
B-2.7	—	B-2.7
B-2.10		B-2.10
B-3.45		B-3.45
B-3.47		B-3.47
8	0	8

FIG.11



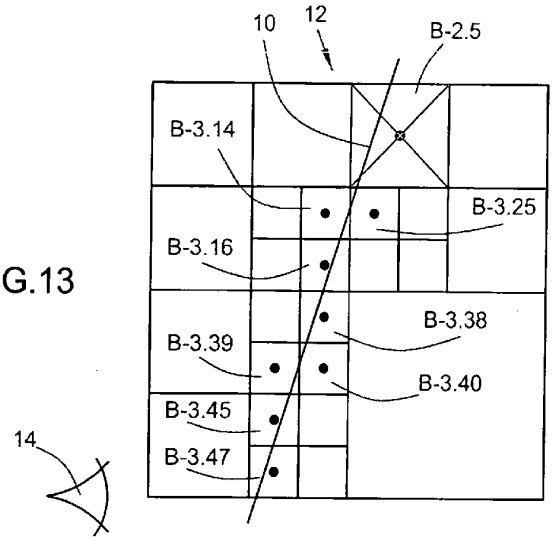
Blocks required	in the memory	to be loaded
B-2.4		B-2.4
B-2.5		B-2.5
B-2.7	—	B-2.7
B-2.10		B-2.10
B-2.12		B-2.12
5	0	5

FIG.12



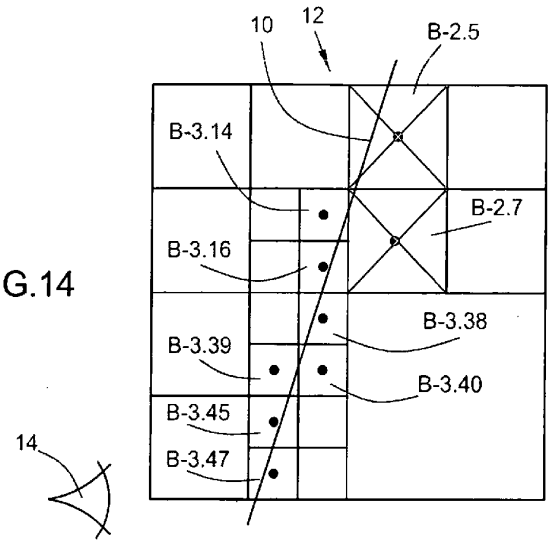
Blocks required	in the memory	to be loaded
B-3.14		B-3.14
B-3.16		B-3.16
B-3.17		B-3.17
B-3.19		B-3.19
B-3.25	—	B-3.25
B-3.38		B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
10	0	10

FIG.13



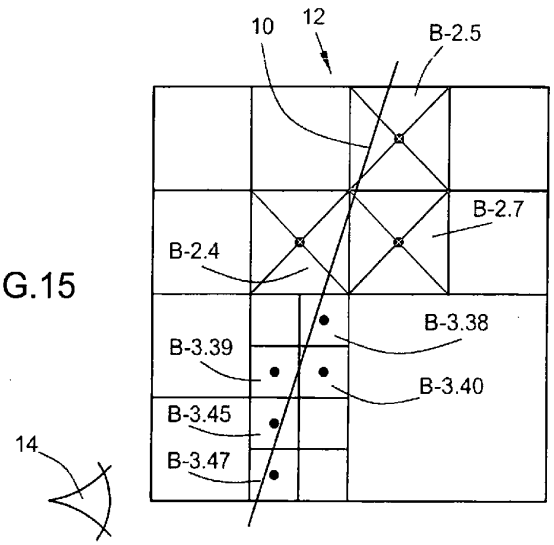
Blocks required	in the memory	to be loaded
B-2.5	B-2.5	
B-3.14		B-3.14
B-3.16		B-3.16
B-3.25		B-3.25
B-3.38		B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
9	1	8

FIG.14



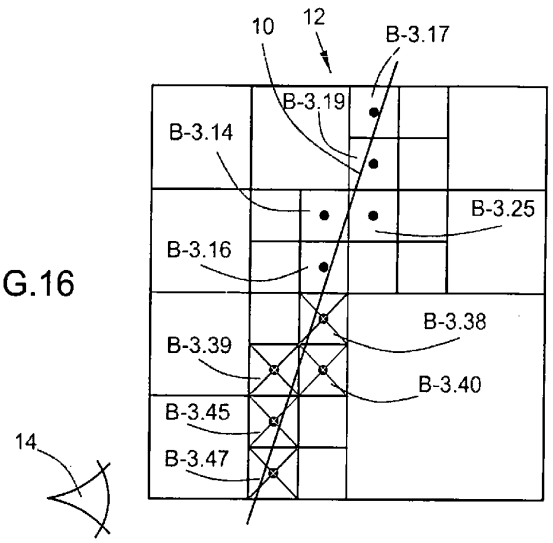
Blocks required	in the memory	to be loaded
B-2.5	B-2.5	
B-2.7	B-2.7	
B-3.14		B-3.14
B-3.16		B-3.16
B-3.38		B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
9	2	7

FIG.15



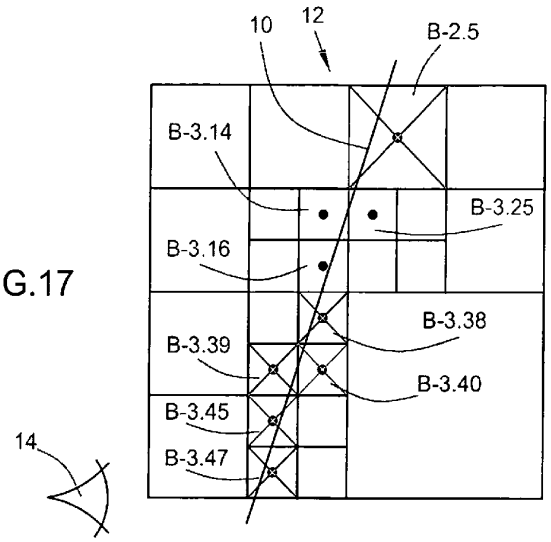
Blocks required	in the memory	to be loaded
B-2.4	B-2.4	
B-2.5	B-2.5	
B-2.7	B-2.7	
B-3.38		B-3.38
B-3.39		B-3.39
B-3.40		B-3.40
B-3.45		B-3.45
B-3.47		B-3.47
8	3	5

FIG.16



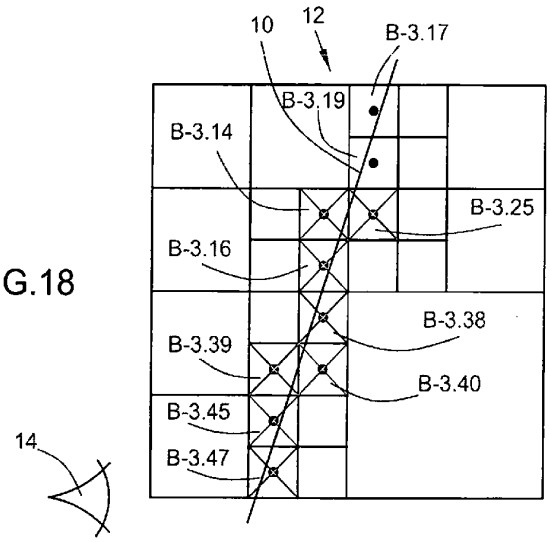
Blocks required	in the memory	to be loaded
B-3.14		B-3.14
B-3.16		B-3.16
B-3.17		B-3.17
B-3.19		B-3.19
B-3.25		B-3.25
B-3.38	B-3.38	
B-3.39	B-3.39	
B-3.40	B-3.40	
B-3.45	B-3.45	
B-3.47	B-3.47	
10	5	5

FIG.17



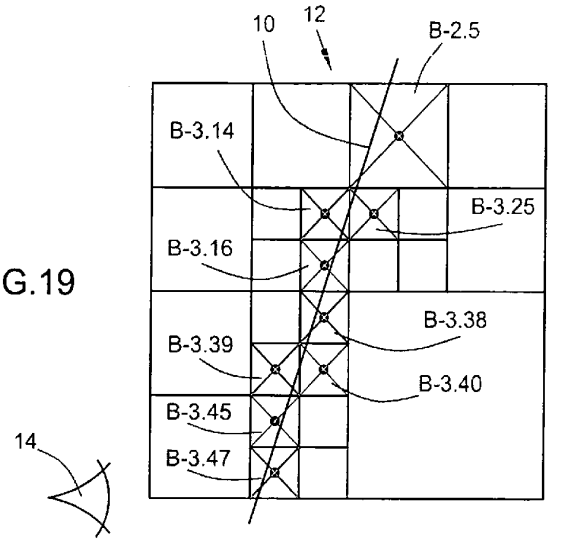
Blocks required	in the memory	to be loaded
B-2.5	B-2.5	
B-3.14		B-3.14
B-3.16		B-3.16
B-3.25		B-3.25
B-3.38	B-3.38	
B-3.39	B-3.39	
B-3.40	B-3.40	
B-3.45	B-3.45	
B-3.47	B-3.47	
9	6	3

FIG.18



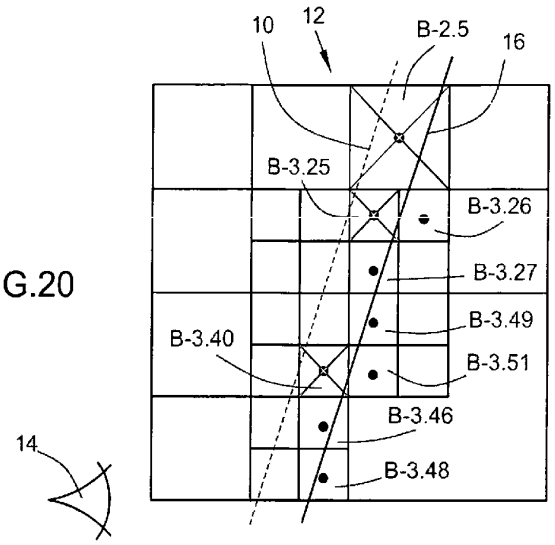
Blocks required	in the memory	to be loaded
B-3.14	B-3.14	
B-3.16	B-3.16	
B-3.17		B-3.17
B-3.19		B-3.19
B-3.25	B-3.25	
B-3.38	B-3.38	
B-3.39	B-3.39	
B-3.40	B-3.40	
B-3.45	B-3.45	
B-3.47	B-3.47	
10	8	2

FIG.19



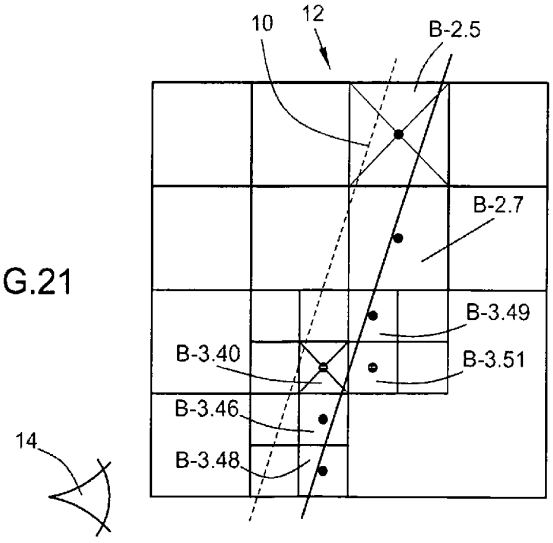
Blocks required	in the memory	to be loaded
B-2.5	B-2.5	
B-3.14	B-3.14	
B-3.16	B-3.16	
B-3.25	B-3.25	
B-3.38	B-3.38	—
B-3.39	B-3.39	
B-3.40	B-3.40	
B-3.45	B-3.45	
B-3.47	B-3.47	
9	9	0

FIG.20



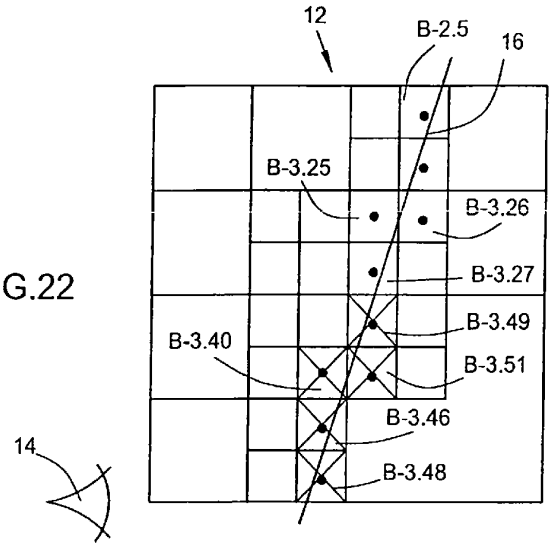
Blocks required	in the memory	to be loaded
9	3	6

FIG.21



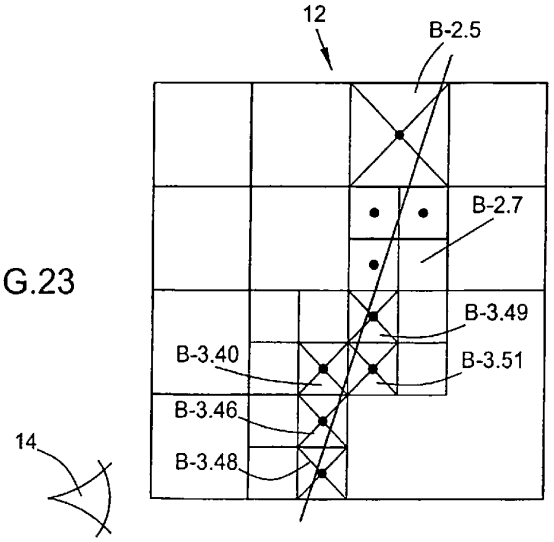
Blocks required	in the memory	to be loaded
7	2	5

FIG.22



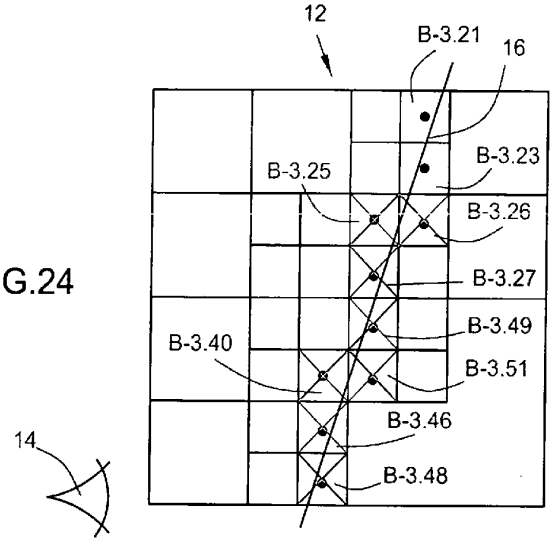
Blocks required	in the memory	to be loaded
10	5	5

FIG.23



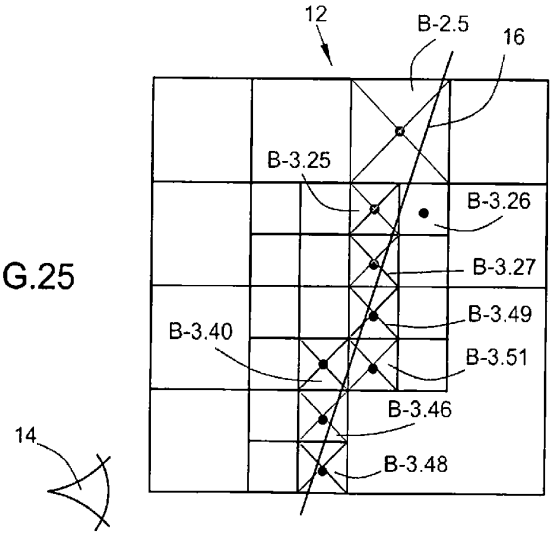
Blocks required	in the memory	to be loaded
9	6	3

FIG.24



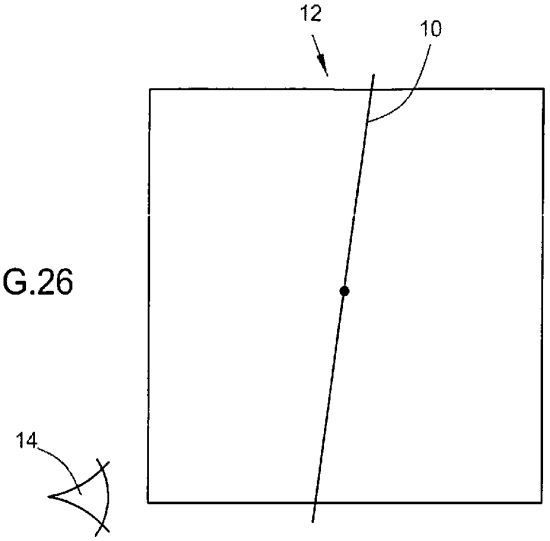
Blocks required	in the memory	to be loaded
10	8	2

FIG.25



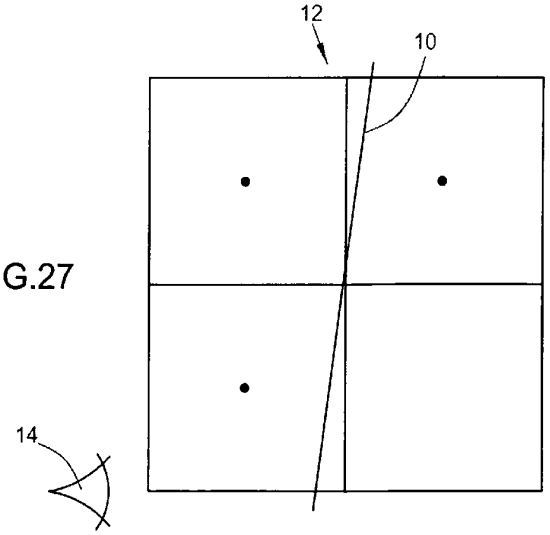
Blocks required	in the memory	to be loaded
9	9	0

FIG.26



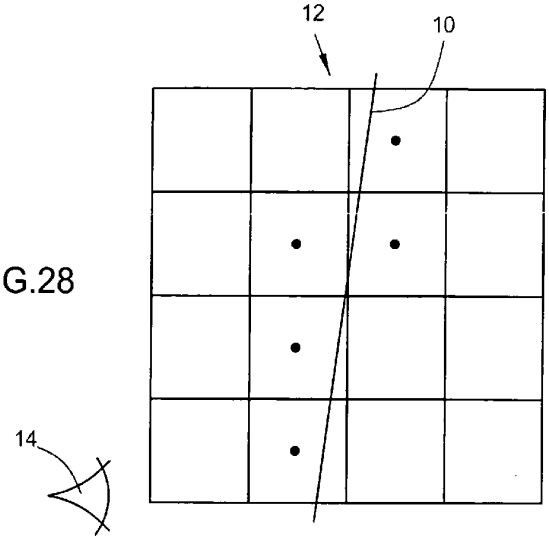
Blocks required	in the memory	to be loaded
1	0	1

FIG.27



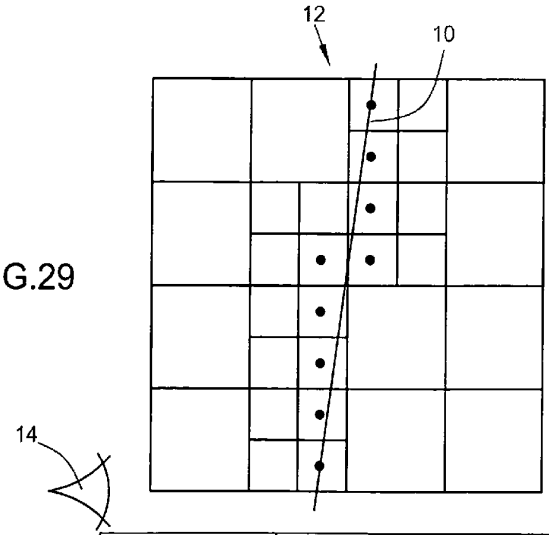
Blocks required	in the memory	to be loaded
3	0	3

FIG.28



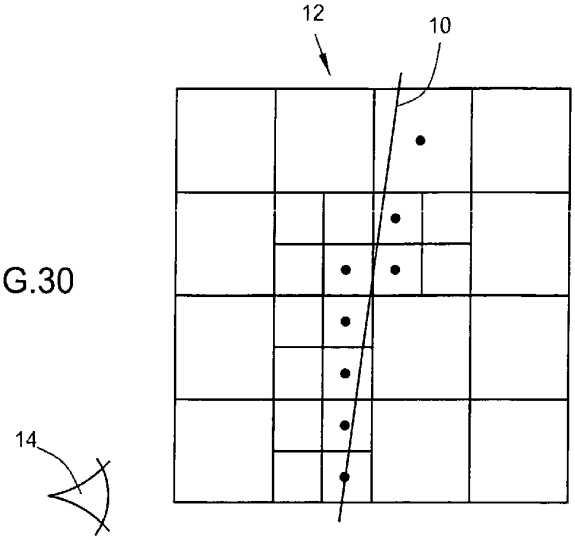
Blocks required	in the memory	to be loaded
5	0	5

FIG.29



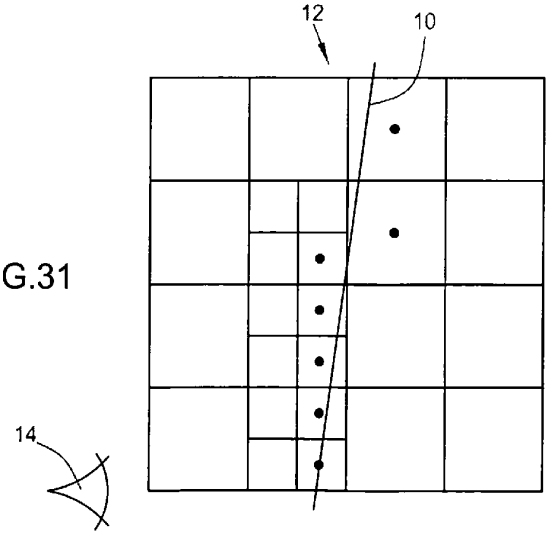
Blocks required	in the memory	to be loaded
9	0	9

FIG.30



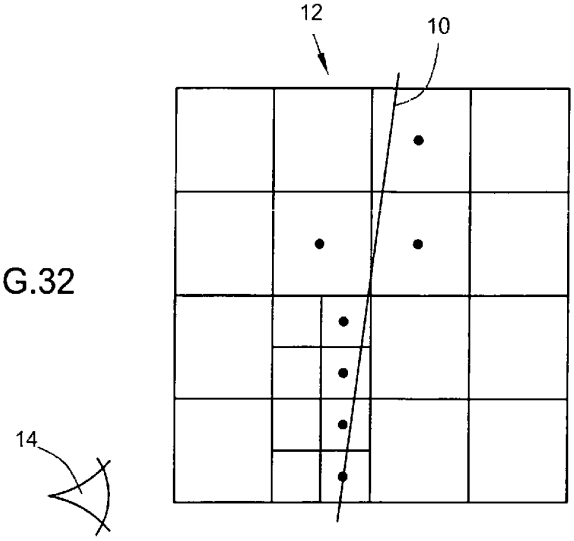
Blocks required	in the memory	to be loaded
8	0	8

FIG.31



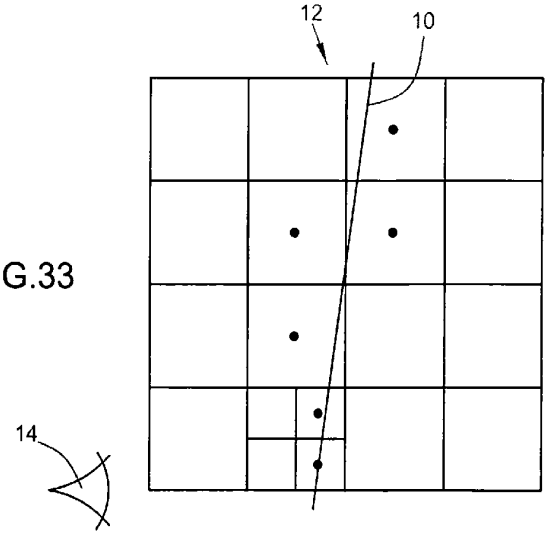
Blocks required	in the memory	to be loaded
7	0	7

FIG.32



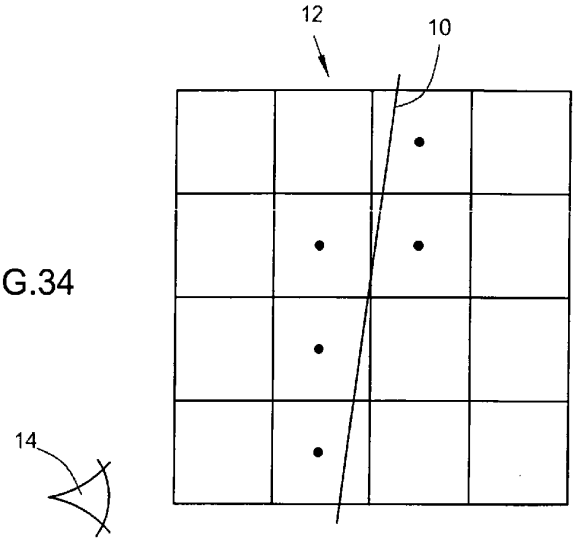
Blocks required	in the memory	to be loaded
7	0	7

FIG.33



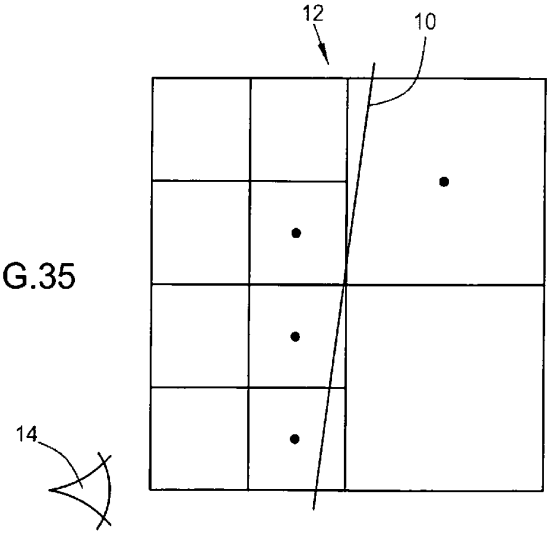
Blocks required	in the memory	to be loaded
6	0	6

FIG.34



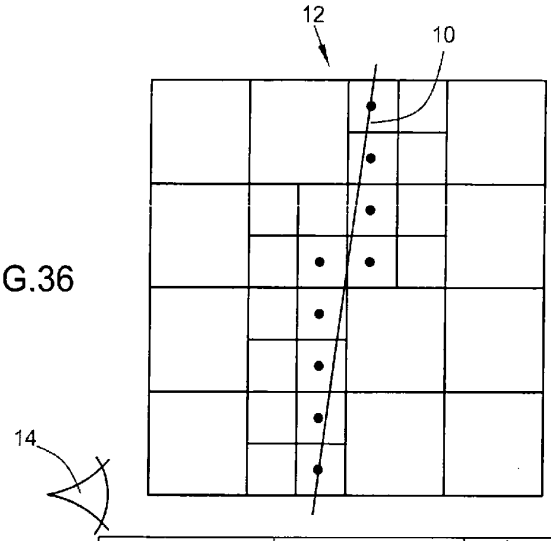
Blocks required	in the memory	to be loaded
5	0	5

FIG.35



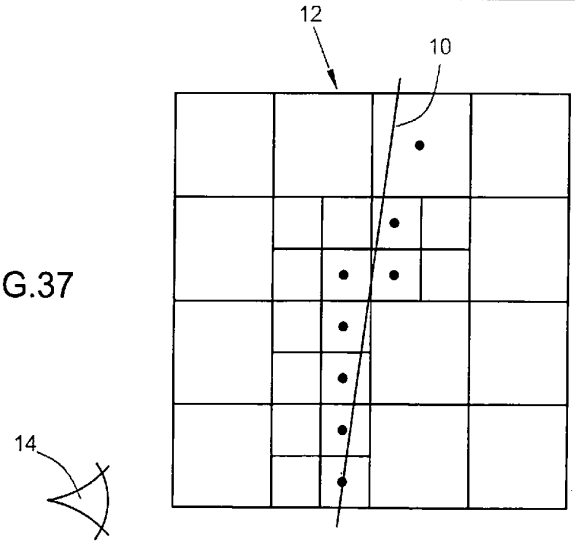
Blocks required	in the memory	to be loaded
4	0	4

FIG.36



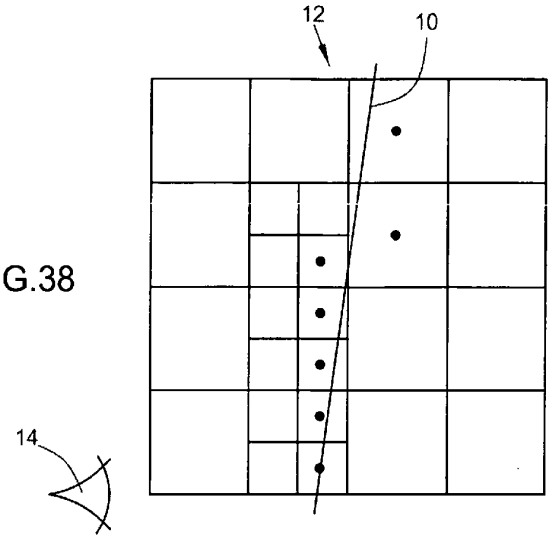
Blocks required	in the memory	to be loaded
9	0	9

FIG.37



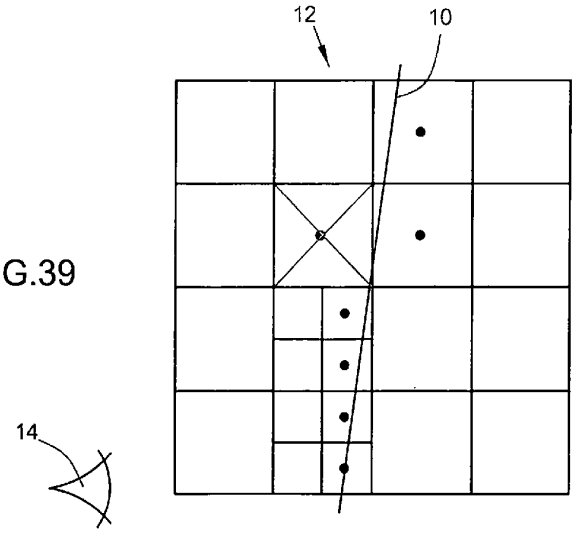
Blocks required	in the memory	to be loaded
8	0	8

FIG.38



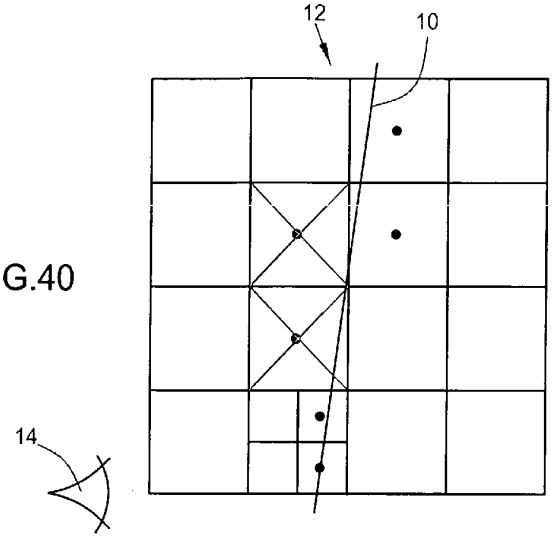
Blocks required	in the memory	to be loaded
7	0	7

FIG.39



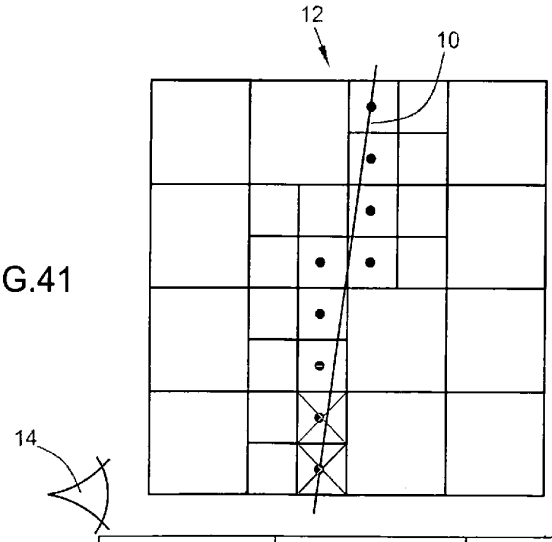
Blocks required	in the memory	to be loaded
7	1	6

FIG.40



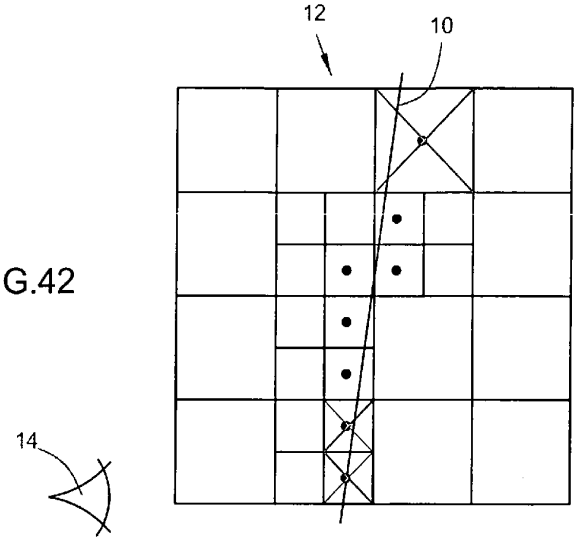
Blocks required	in the memory	to be loaded
6	2	4

FIG.41



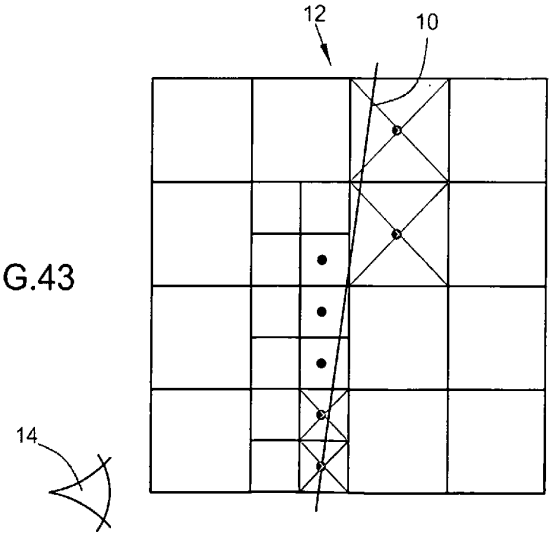
Blocks required	in the memory	to be loaded
9	2	7

FIG.42



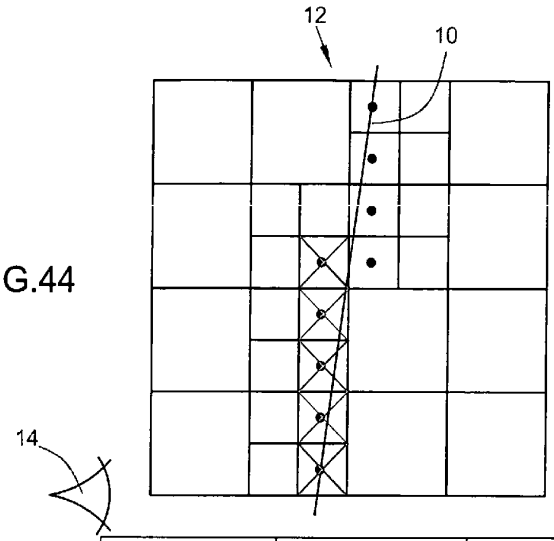
Blocks required	in the memory	to be loaded
8	3	5

FIG.43



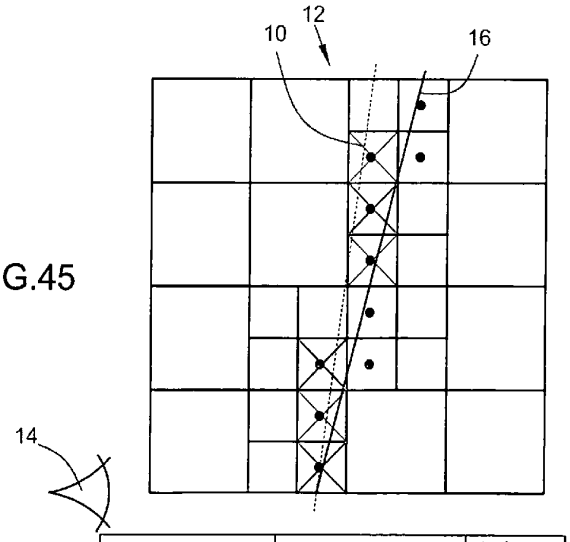
Blocks required	in the memory	to be loaded
7	4	3

FIG.44



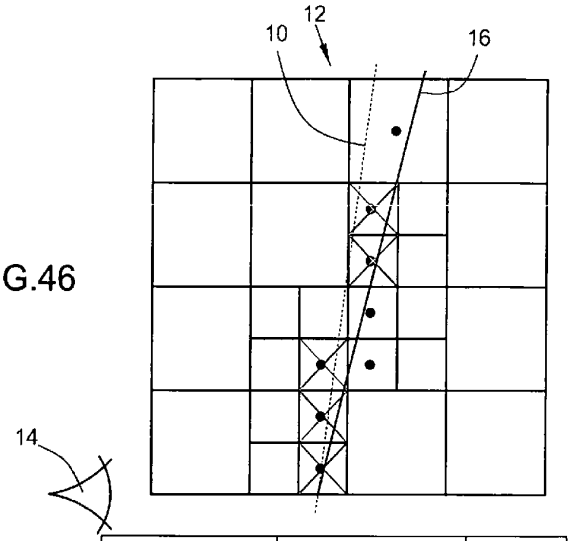
Blocks required	in the memory	to be loaded
9	5	4

FIG.45



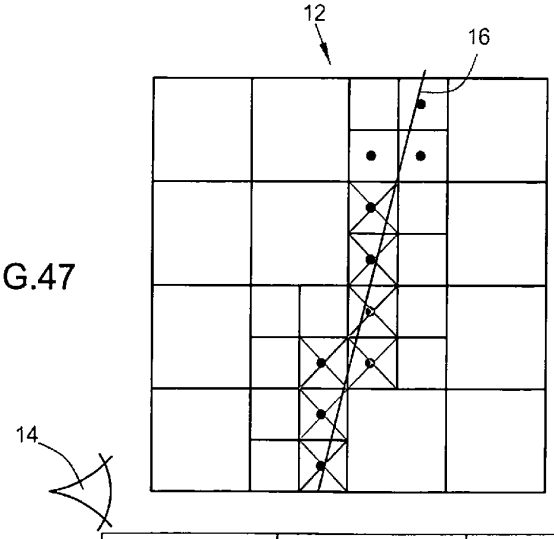
Blocks required	in the memory	to be loaded
10	6	4

FIG.46



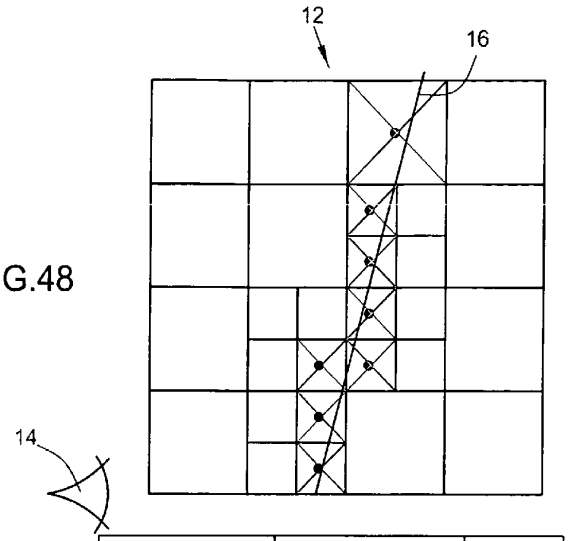
Blocks required	in the memory	to be loaded
8	5	3

FIG.47



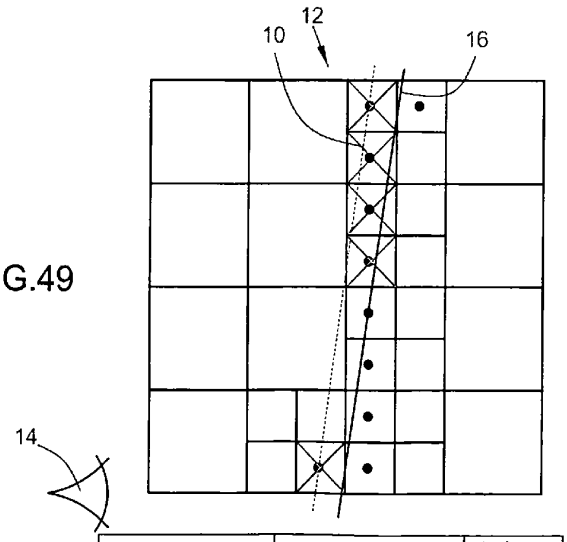
Blocks required	in the memory	to be loaded
10	7	3

FIG.48



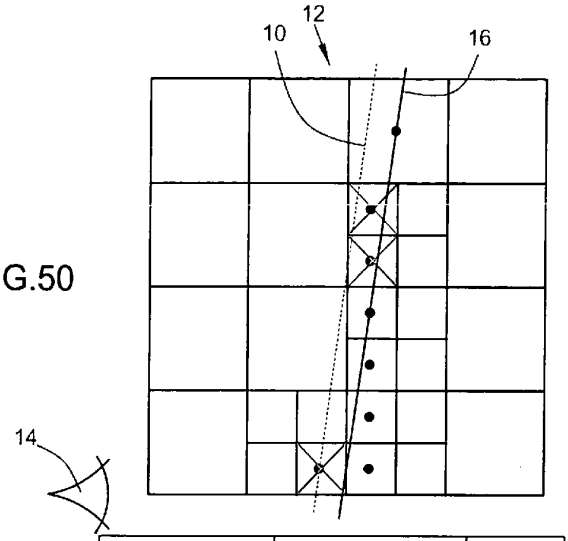
Blocks required	in the memory	to be loaded
8	8	0

FIG.49



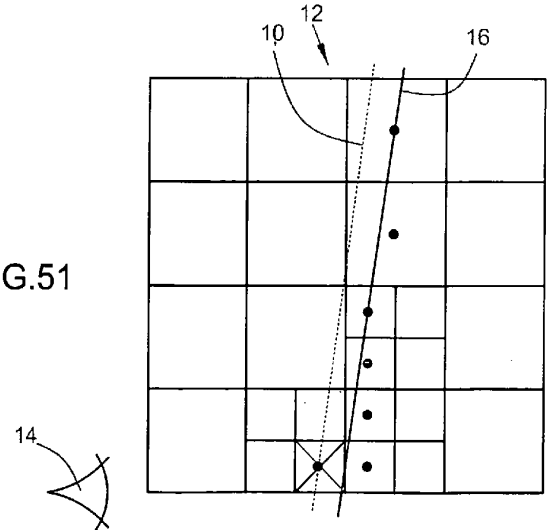
Blocks required	in the memory	to be loaded
10	5	5

FIG.50



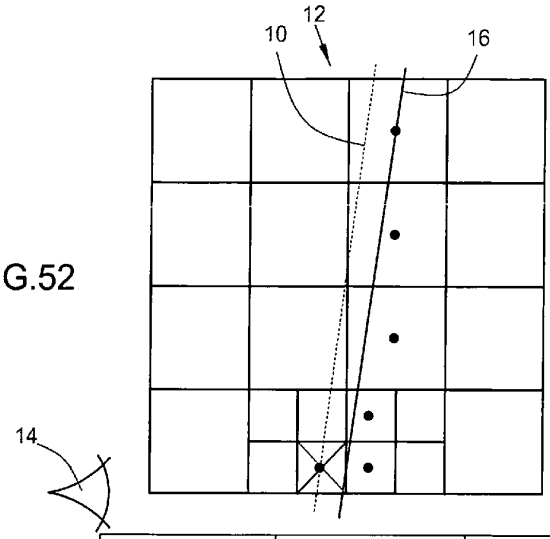
Blocks required	in the memory	to be loaded
8	3	5

FIG.51



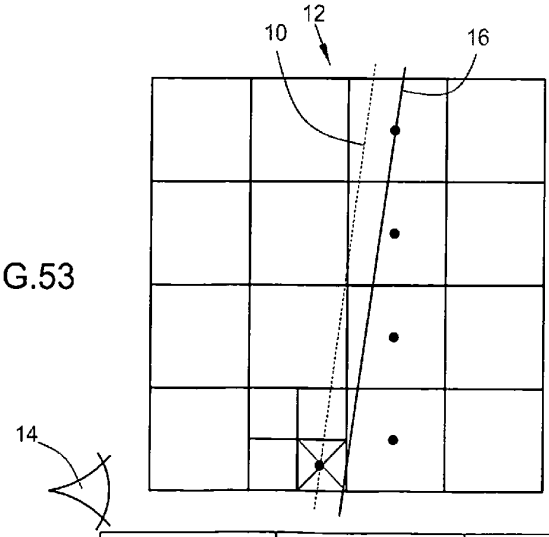
Blocks required	in the memory	to be loaded
7	1	6

FIG.52



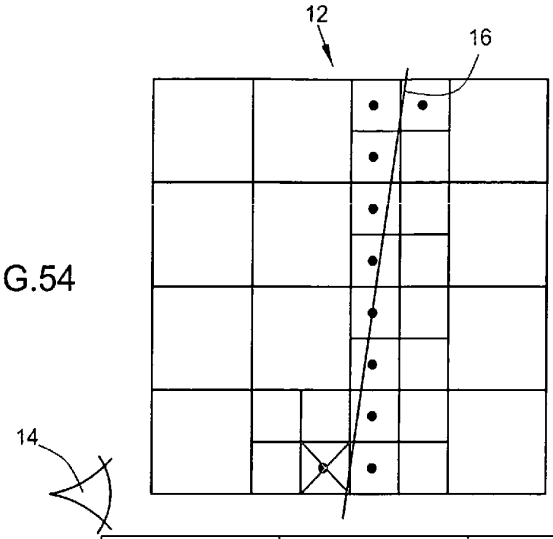
Blocks required	in the memory	to be loaded
6	1	5

FIG.53



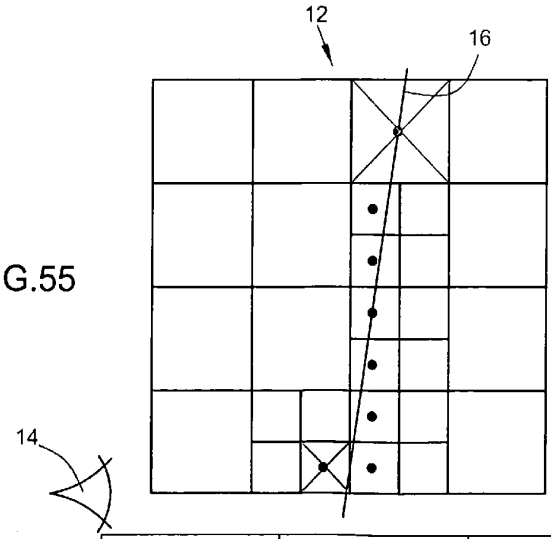
Blocks required	in the memory	to be loaded
5	1	4

FIG.54



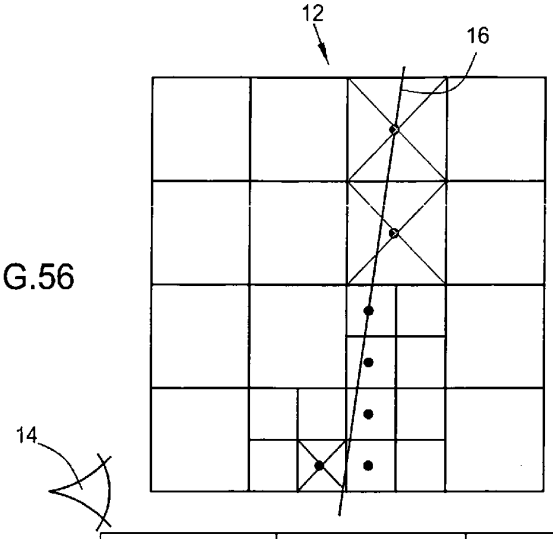
Blocks required	in the memory	to be loaded
10	1	9

FIG.55



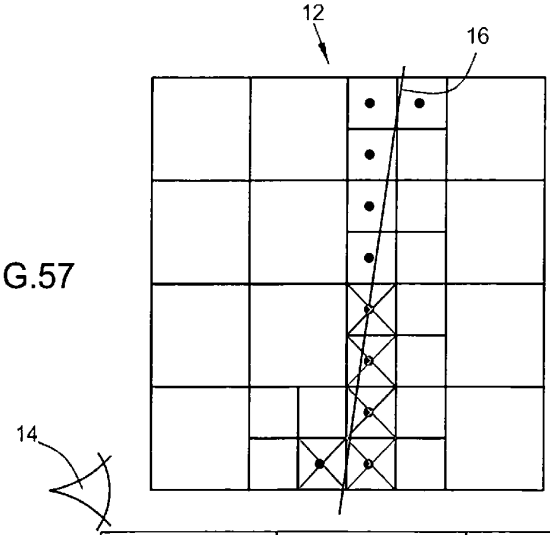
Blocks required	in the memory	to be loaded
8	2	6

FIG.56



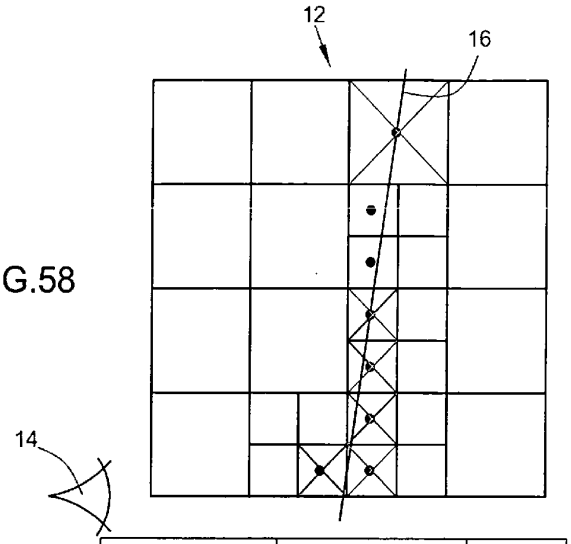
Blocks required	in the memory	to be loaded
7	3	4

FIG.57



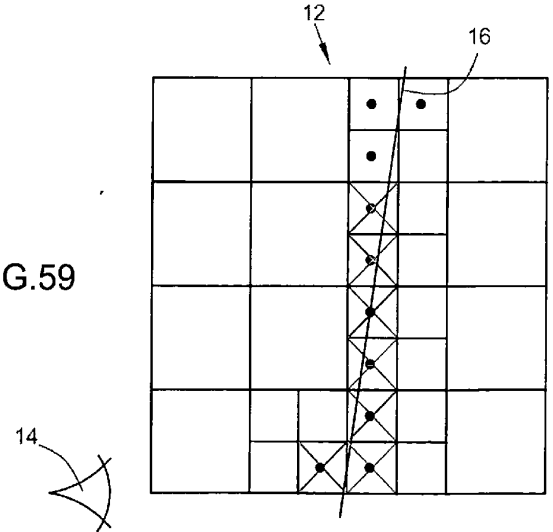
Blocks required	in the memory	to be loaded
10	5	5

FIG.58



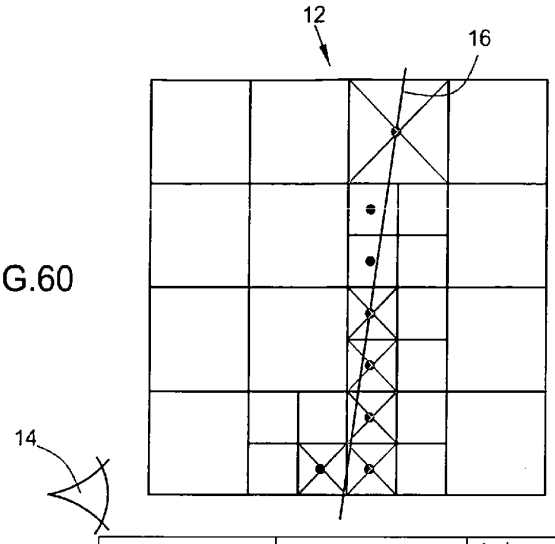
Blocks required	in the memory	to be loaded
8	6	2

FIG.59



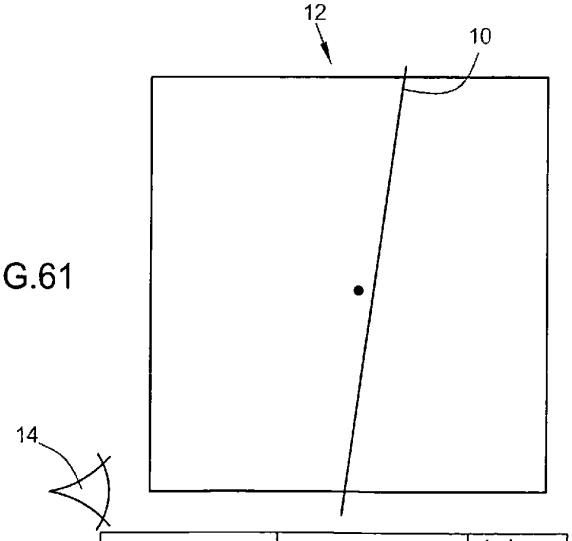
Blocks required	in the memory	to be loaded
10	7	3

FIG.60



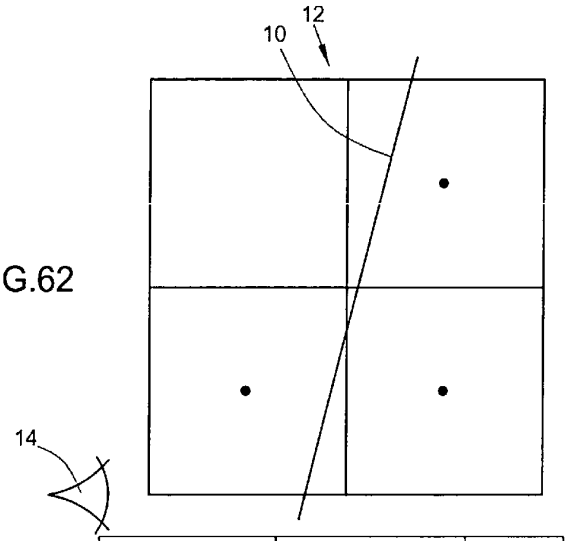
Blocks required	in the memory	to be loaded
8	6	2

FIG.61



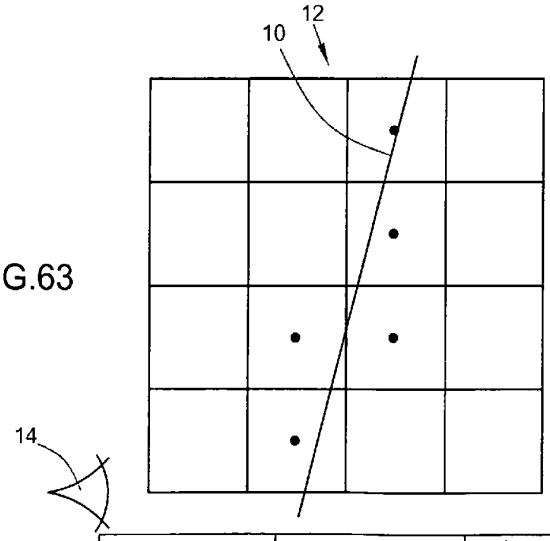
Blocks required	in the memory	to be loaded
1	0	1

FIG.62



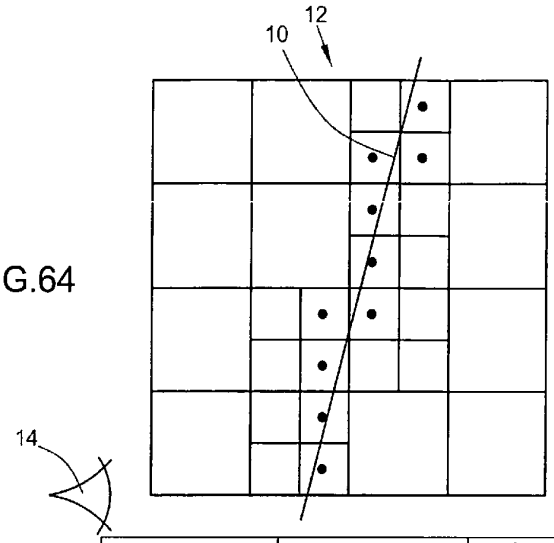
Blocks required	in the memory	to be loaded
3	0	3

FIG.63



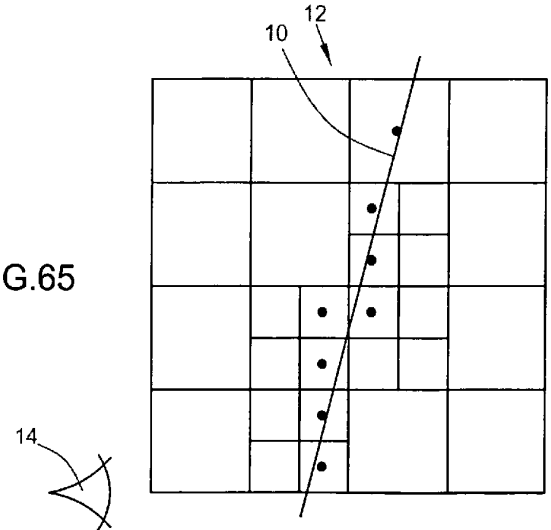
Blocks required	in the memory	to be loaded
5	0	5

FIG.64



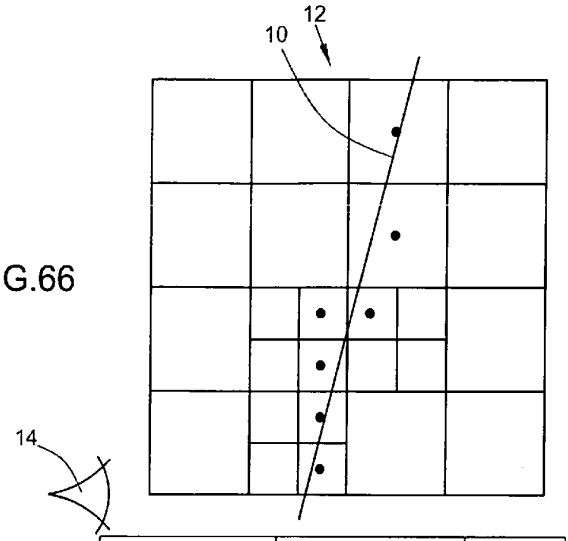
Blocks required	in the memory	to be loaded
10	0	10

FIG.65



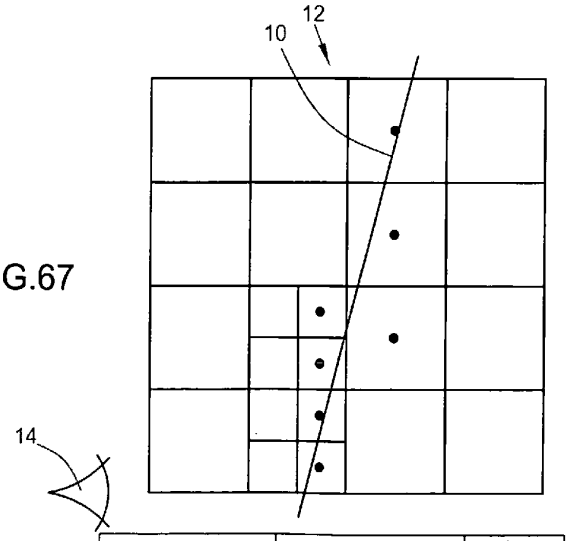
Blocks required	in the memory	to be loaded
8	0	8

FIG.66



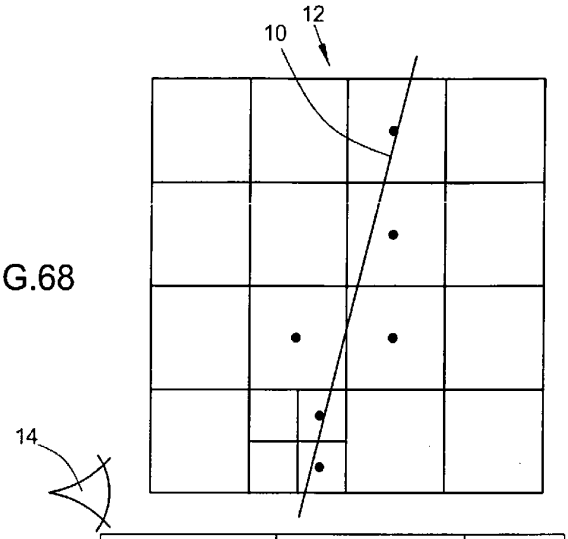
Blocks required	in the memory	to be loaded
7	0	7

FIG.67



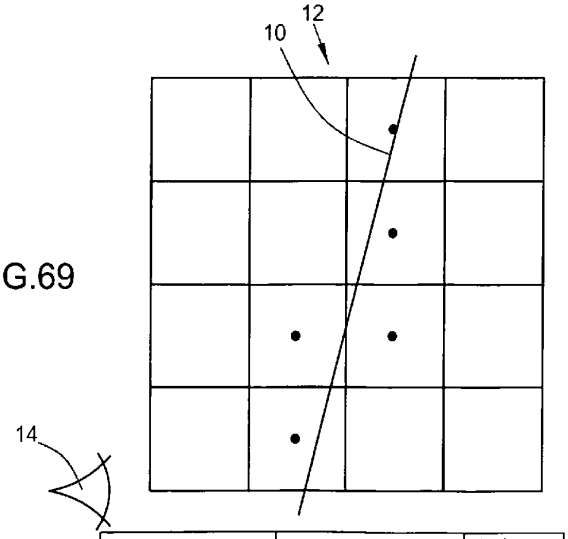
Blocks required	in the memory	to be loaded
7	0	7

FIG.68



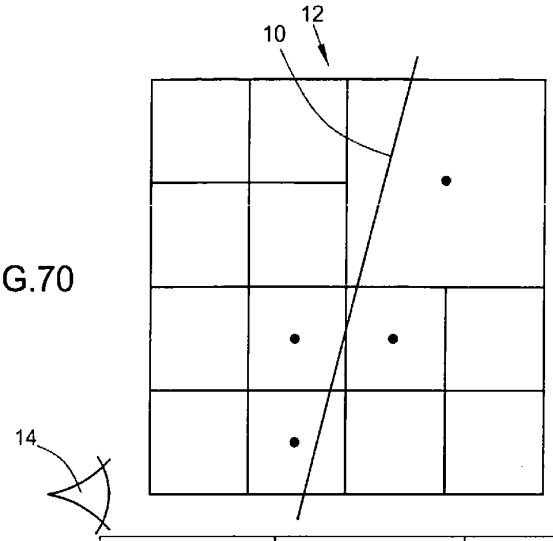
Blocks required	in the memory	to be loaded
6	0	6

FIG.69



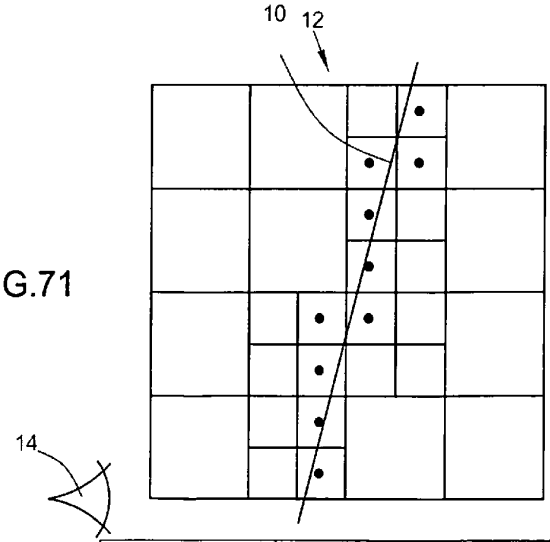
Blocks required	in the memory	to be loaded
5	0	5

FIG.70



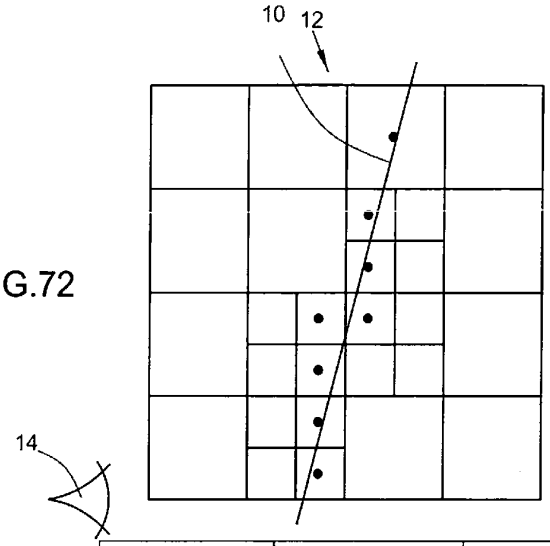
Blocks required	in the memory	to be loaded
4	0	4

FIG.71



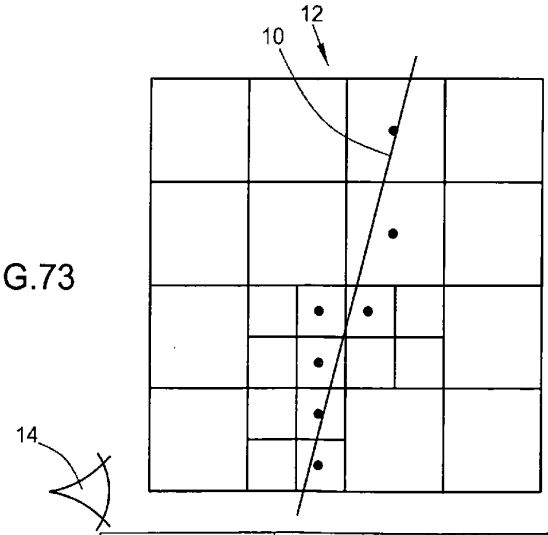
Blocks required	in the memory	to be loaded
10	0	10

FIG.72



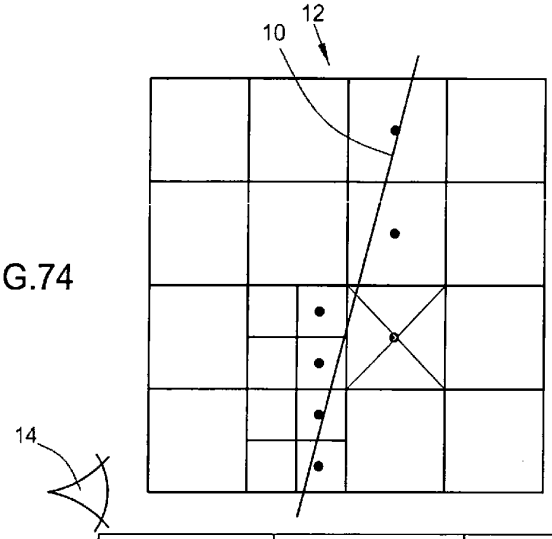
Blocks required	in the memory	to be loaded
8	0	8

FIG.73



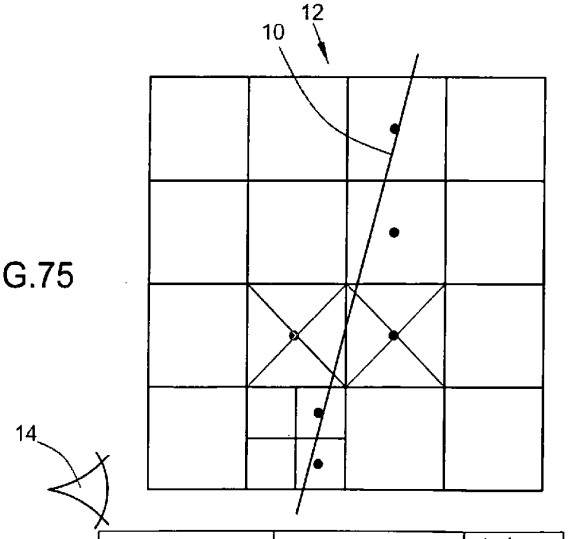
Blocks required	in the memory	to be loaded
7	0	7

FIG.74



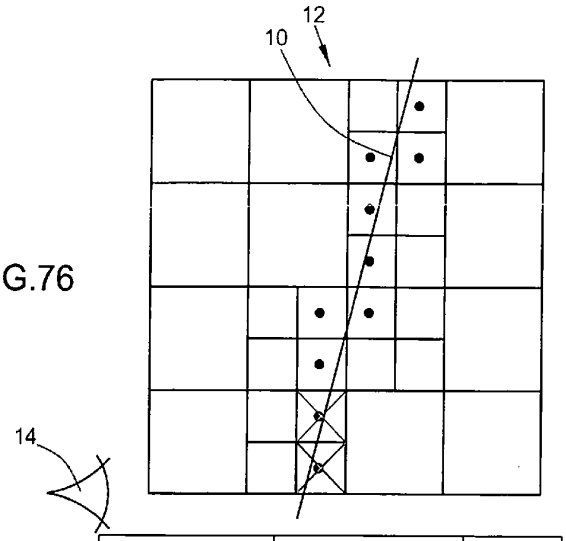
Blocks required	in the memory	to be loaded
7	1	6

FIG.75



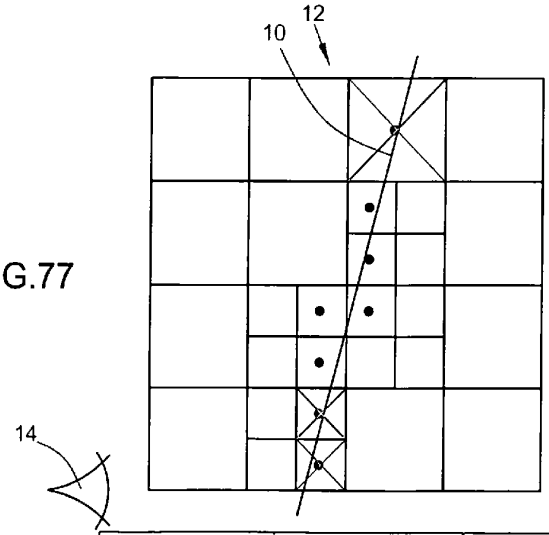
Blocks required	in the memory	to be loaded
6	2	4

FIG.76



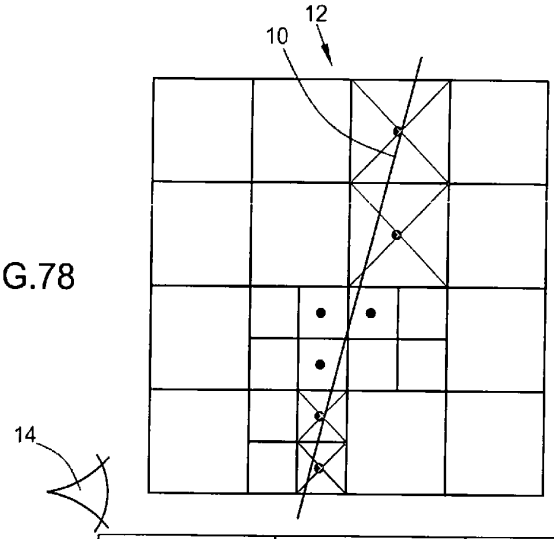
Blocks required	in the memory	to be loaded
10	2	8

FIG.77



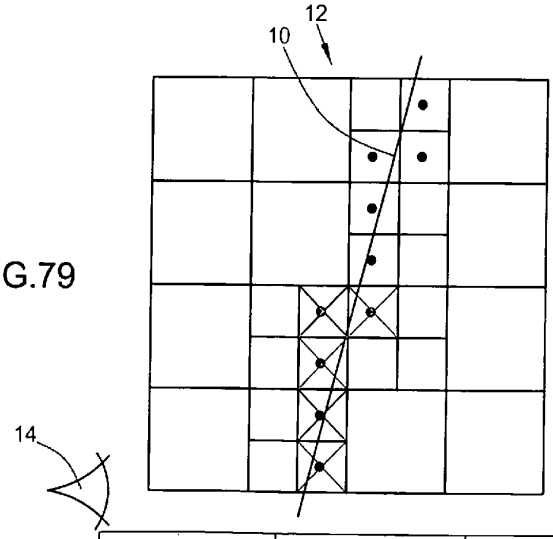
Blocks required	in the memory	to be loaded
8	3	5

FIG.78



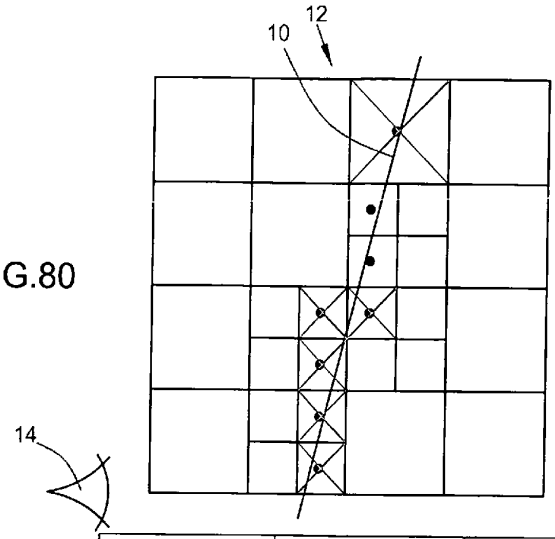
Blocks required	in the memory	to be loaded
7	4	3

FIG.79



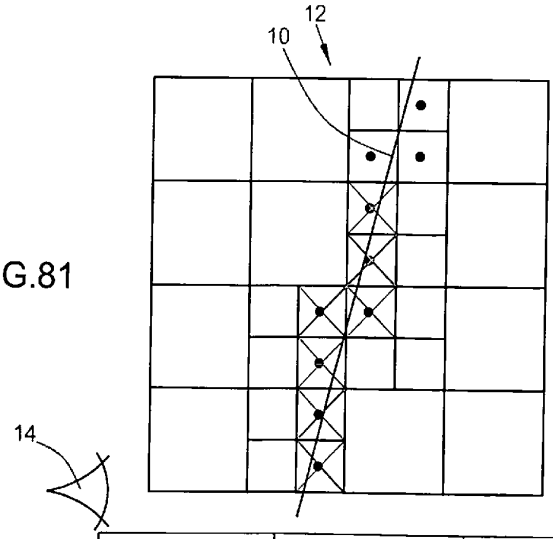
Blocks required	in the memory	to be loaded
10	5	5

FIG.80



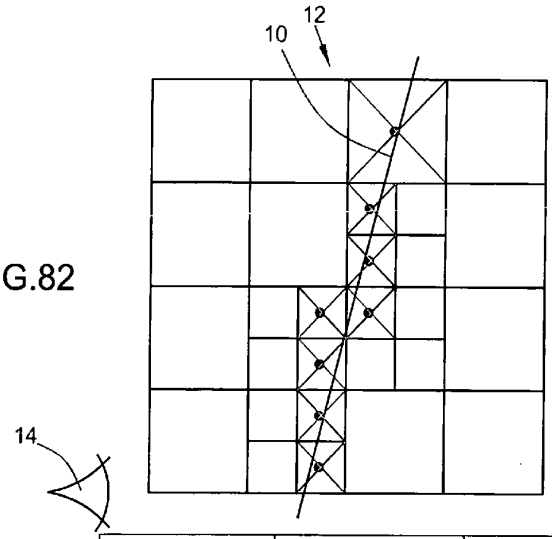
Blocks required	in the memory	to be loaded
8	6	2

FIG.81



Blocks required	in the memory	to be loaded
10	7	3

FIG.82



Blocks required	in the memory	to be loaded
8	8	0