(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 January 2002 (24.01.2002)

PCT

(10) International Publication Number
WO 02/07088 A2

(51) International Patent Classification⁷: G06T

(21) International Application Number: PCT/CA01/01010

(22) International Filing Date: 10 July 2001 (10.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2,315,302 13 July 2000 (13.07.2000) CA

(71) Applicant *(for all designated States except US)*: **i3DIMENSIONS INC.** [CA/CA]; Suite 205, 1818 Cornwall Avenue, Vancouver, British Columbia V6J 1C7 (CA).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **HALMSHAW, Paul, A.** [CA/CA]; 245 West 19th Street, North Vancouver, British Columbia V7M 1X6 (CA).

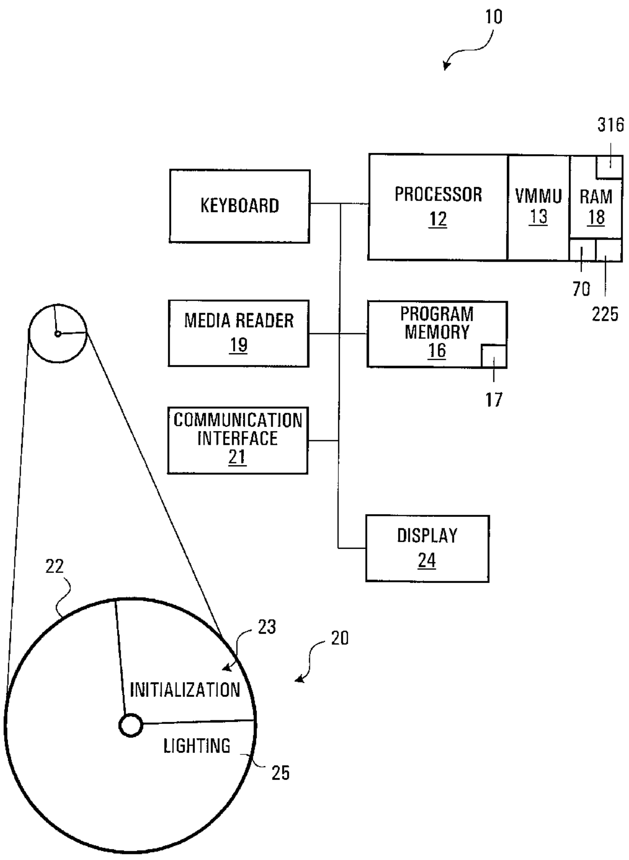
(74) Agents: **KNOX, John, W.** et al.; Fetherstonhaugh & Co., Box 11560, Vancouver Centre, Suite 2200, 650 West Georgia Street, Vancouver, British Columbia V6B 4N8 (CA).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: LIGHTING ADJUSTMENT METHODS AND APPARATUS FOR VOXEL DATA



(57) **Abstract:** A method and apparatus for producing a lighting effect on voxel data is disclosed. The method involves computing specular and diffuse illumination values for respective vectors of a set of mormal vectors representing a plurality of possible different directions away from any voxel. The specular and diffuse illumination values are then associated with respective normal vectors to facilitate retrieval of the specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced. The method may further involve identifying a voxel upon which lighting effects are to be produced and modifying prestored lighting values associated with that voxel, according to at least one of the specular and diffuse illumination values determined using a normal vector of the identified voxel.

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

LIGHTING ADJUSTMENT METHODS AND APPARATUS FOR VOXEL DATA

BACKGROUND OF THE INVENTION

1. Field of Invention

This invention relates generally to computer graphic systems and more particularly to methods and apparatus for creating lighting effects for voxel data.

2. Description of Related Art

With the increasing number of computer games and imaging technologies, the demand for improved computer graphics capabilities is increased. Computer graphic technologies have had to make compromises between picture quality, memory usage and speed to meet the needs of new games and new and increasing usage of computer graphics.

Some technologies have provided superior picture quality by meticulous computations on every voxel in a voxel object, to apply lighting effects. Generally, various effects algorithms are applied to each and every voxel as voxels are rendered. For example some algorithms may employ floating point arithmetic to calculate for each voxel its normal and/or intensity value based on specular and diffuse illumination components derived from the relative positions of the voxel object, a view plane and a lighting plane. In addition some algorithms may employ complex time-intensive sorting algorithms to determine the relative positions of voxels with respect to the lighting plane and the view plane to determine which voxels are occluded with respect to the light and/or with respect to the view plane. This can slow down the rendering process, resulting in a complete render of the voxel object being unacceptably slow.

To increase rendering speed, more memory can be used or compromises can be made in the algorithms, at the expense of picture quality. However, these expenses are desirably kept to a minimum.

What would be desirable therefore is a way of increasing the speed of rendering without significant loss of picture quality or a significant increase in memory usage.

SUMMARY OF THE INVENTION

In accordance with one aspect of the invention, there is provided a method of producing a lighting effect on voxel data. The method involves computing specular and diffuse illumination values for respective vectors of a set of normal vectors representing a plurality of possible different directions away from any voxel. Then the specular and diffuse illumination values are associated with respective normal vectors to facilitate retrieval of the specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced.

The method may further involve computing the direction of a half-way vector H at an angle half-way between an illumination vector L representing a direction to a lighting plane and a viewing vector V representing a direction to a view plane.

The method may involve producing a table associating a plurality of discrete normal vectors with corresponding specular illumination values calculated according to the relation:

$$S = (N \bullet H)^k$$

Where: S is the calculated specular illumination value

N is a discrete unit normal vector from the set of normal vectors

H is a unit half-way vector between a unit view vector V and a unit lighting vector L
k is an exponent representing a property of a surface

The method may also involve producing a table associating a plurality of discrete normal vectors with corresponding diffuse illumination values calculated according to the relation:

$$D = N \bullet L$$

Where: D is the calculated specular illumination value
N is a discrete unit normal vector from the set of normal vectors
L is a unit illumination vector

The production of specular and diffuse illumination values may involve negating L when the dot product of normalized V and L vectors is negative.

Producing the tables may involve apportioning the specular and diffuse illumination tables into portions associated with respective areas of a sphere within which the normal vectors may lie.

The method may further involve identifying a voxel upon which lighting effects are to be produced and modifying prestored lighting values associated with that voxel, according to at least one of the specular and diffuse illumination values determined using a normal vector of the identified voxel.

The method may further involve identifying potentially illuminable voxels in the voxel block. Such identifying may involve scanning the voxel block to produce position specifiers identifying the potentially illuminable voxels.

The position specifiers may be mapped into lighting plane positions and a reference identifying a voxel specified by the voxel position specifier, may be associated with a lighting plane position determined by the mapping, when a reference has not already been associated with the lighting plane position.

A priority table may be loaded to associate lighting plane positions with corresponding references. The priority table may be addressed to find a priority reference in the priority table, associated with the lighting plane position.

The method may further involve modifying lighting values associated with voxels identified by the priority references in the priority table.

The specular and diffuse illumination values may be used to change an intensity value associated with a voxel identified by a priority reference in the priority table.

Modifying the lighting values may comprise locating the specular and diffuse illumination values in response to a normal vector associated with the voxel identified by the priority reference. Locating may involve using the normal vector as an index to a table associating normal vectors with the specular and diffuse illumination values.

The method may further involve receiving a reference identifying a voxel upon which a lighting effect is to be produced and mapping a voxel identified by a received reference into a lighting plane position.

The method may involve modifying a lighting intensity value associated with the voxel identified by the priority reference, using the specular and diffuse lighting values, when the priority reference associated with the lighting plane matches the received reference.

In accordance with another aspect of the invention, there is provided a method of producing a lighting effect on voxel data. The method may involve identifying renderable voxels of a voxel block that are more foreground than other renderable voxels in the voxel block, relative to a lighting plane, and modifying lighting values associated with identified voxels, according to pre-computed specular and diffuse illumination values associated with normal vectors of the voxels.

In accordance with another aspect of the invention, there is provided a method of associating lighting plane positions on a lighting plane with voxels in a voxel block, for use in determining which voxels are to receive lighting effects. The method may involve scanning voxel positions in the voxel block, according to a scan order in which column positions, in respective planes through the voxel block that are more foreground relative to the lighting plane, are scanned before those that are more background relative to the lighting plane, to produce position specifiers identifying illuminable voxel positions in the voxel block. Then, respective lighting plane positions for the position specifiers may be determined according to a lighting plane position mapping. Respective lighting plane positions may then be associated with references to voxels identified by the position specifiers such that the lighting plane positions that are already associated with references do not receive new references when a position specifier maps to a lighting plane position to which a reference has already been mapped. A record of the lighting plane positions and their associated references identifying voxels to which lighting effects may be applied may be maintained for use by lighting effects programs or routines.

In accordance with another aspect of the invention, there is provided an apparatus for producing a lighting effect on voxel data. The apparatus comprises a processor circuit and memory. The processor circuit is operable to compute specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel in a voxel block. The memory is accessible by the processor circuit to enable the processor circuit to associate the specular and diffuse illumination values with respective unit normal vectors to facilitate retrieval of the specular and diffuse illumination values associated with a unit normal vector of a voxel upon which the lighting effect is to be produced.

The processor circuit may be programmed to compute a direction of a unit half-way vector H at an angle half-way between a unit illumination vector L representing a direction to a lighting plane and a unit viewing vector V representing a direction to a view plane.

The processor circuit may also be programmed to compute a table associating a plurality of quantized unit normal vectors with corresponding specular illumination values calculated according to the relation:

$$S = (N \bullet H)^k$$

Where S is a calculated specular illumination value

N is a quantized unit normal vector from the set of unit normal vectors

H is the unit half-way vector between the unit viewing vector V and the unit illumination vector L, and

k is an exponent representing a property of a surface,

The processor circuit may also be programmed to produce a table associating a plurality of quantized unit normal vectors with corresponding diffuse illumination values calculated according to the relation:

$$D = N \bullet L$$

Where D is a calculated specular illumination value

N is a quantized unit normal vector from the set of unit normal vectors; and
L is the unit illumination vector,

The processor circuit may produce the diffuse and specular illumination values by negating L when the dot product of the V and L vectors is negative.

The processor circuit may be operable to apportion the tables into portions associated with respective areas of a sphere within which the unit normal vectors may lie.

The apparatus may further comprise an identifier in communication with the processor circuit and operable to identify a voxel upon which the lighting effect is to be produced. The identifier may include the processor circuit.

The apparatus may further comprise a modifier operable to communicate with the memory to obtain the specular and diffuse illumination values and operable to access and modify prestored lighting values associated with an identified voxel according to at least one of the specular and diffuse illumination values determined using a unit normal vector of the identified voxel. The modifier may comprise the processor circuit.

The identifier may further comprise a scanner operable to scan the voxel block and may be operable to associate a reference identifying a voxel specified by a voxel position specifier, with a lighting plane position determined by a mapper, when a reference has not already been associated with the lighting plane position.

The apparatus may further comprise a mapper for mapping the position specifiers produced by the identifier into lighting plane positions.

The scanner may be operable to scan the voxel block according to a scan order in which column positions in respective planes through the voxel block that are progressively farther from a lighting plane are addressed in order of those that map to foreground positions in the lighting plane before those which map to more background positions in the lighting plane, to produce the voxel position specifiers identifying potentially illuminable voxels. The scanner may comprise the processor circuit.

The apparatus may further comprise memory accessible by the scanner for storing a priority table operable to associate lighting plane positions with respective the references, with priority given to voxel references that are in more foreground planes relative to a lighting plane over voxel references to voxels that are in more background planes relative to the lighting plane.

The modifier may be in communication with the memory storing the priority table and may be operable to modify lighting values associated with voxels identified by the references in the priority table.

The modifier may be operable to apply the specular and diffuse illumination values to change an intensity value associated with a voxel identified by a reference in the priority table and may be operable to locate the specular and diffuse illumination values in the memory in response to a unit normal vector associated with the voxel identified by the reference.

The apparatus may further comprise a receiver in communication with the modifier, operable to receive a reference identifying a voxel upon which the lighting effect is to be produced.

Other aspects and features of the present invention will become apparent to those ordinarily skilled in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

In drawings which illustrate embodiments of the invention,

Figure **1**	is a block diagram of a computer system implementing a lighting adjustment apparatus according to a first embodiment of the invention.
Figure **2**	is a schematic representation of a voxel block representing voxel data to which lighting effects are applied by the apparatus of Figure **1**.
Figure **3**	is a schematic representation of a data structure for storing data associated with voxels in the voxel blocks shown in Figure **2**.
Figure **4**	is a schematic representation of a global coordinate system in which the voxel block of Figure **2** is situated, together with an illumination plane and a viewing plane.
Figure **5**	is a flow chart of a lighting program which implements a method according to the first embodiment of the invention.
Figure **6**	is a flow chart of a lighting program according to an alternate embodiment of the invention.
Figure **7**A and B	together depict a flow chart of an initialization routine of the lighting program shown in Figure **5**.
Figure **8**	is a schematic representation of the determination of a lighting plane orientation vector according to the initialization routine shown in Figure **7**A.

Figure **9**	is a schematic representation of the determination of a base point according to the initialization routine shown in Figure **7**A.
Figure **10**	is a schematic representation of the determination of plane and column axes representing a scan order produced in accordance with the intialization routine shown in Figure **7**A.
Figure **11**	is a tabular representation of an address conversion table produced according to the initialization routine shown in Figure **7**A.
Figure **12**	is a schematic representation of a column clipping table which may be employed by a priority routine of the initialization routine, shown in Figure **7**B.
Figure **13**	is a schematic representation of a clipping plane produced by the clipping table shown in Figure **12**.
Figure **14**	is a schematic representation of a clipping plane produced by the clipping range table shown in Figure **12**.
Figure **15**	is a tabular representation of specular and diffuse illumination tables produced by the initialization routine shown in Figure **7**A.
Figure **16**	is a flow chart of a lighting routine which is part of the lighting program shown in Figure **5**.
Figure **17**	is a flow chart of an output routine for providing RGB output intensity values for use in controlling a display.
Figure **18**	is a block diagram of an apparatus according to an alternative embodiment of the invention.

DETAILED DESCRIPTION

Referring to Figure **1**, an apparatus according to one embodiment of the invention is shown generally at **10**. In this embodiment, the apparatus comprises a computer system, such as a personal computer having a processor circuit **12**, program memory **16** and random access memory **18**. The processor circuit **12** has a virtual memory mapping unit (VMMU) **13** that converts logical addresses provided by any application programs running on the processor circuit to physical memory address space in the random access memory **18**. The processor circuit **12** runs an operating system (not shown) that is operable to define cache memory within the random access memory **18** and to work in pages of memory therein.

The computer system **10** may further include a media reader **19** in communication with the processor circuit **12** for enabling the processor circuit to read a set of codes **20** or computer program, from a computer readable medium **22** such as a floppy disk, a hard disk, a CD-ROM, or a communications channel, for example. Alternatively, the set of codes **20** may be provided as segments of a signal embodied in a carrier wave received at a communications interface **21** in communication with the processor circuit **12**, for example. In other words, the set of codes or program may be downloaded from a communications system such as the Internet, into the computer system **10**.

The set of codes **20** implements a lighting program operable to be stored in the program memory **16.** The program directs the processor circuit **12** to carry out functional actions according to a method for producing lighting effects for display of information associated with data of a volume data set stored in a data structure, in accordance with other aspects of the invention. The data structure may be stored in a data structure area **70** of the RAM **18**, for example, or in remotely accessible memory (not shown), remotely accessible by the processor circuit.

The set of codes **20** may contain a plurality of segments including an initialization segment **23** and a lighting segment **25** as will be described below.

The embodiments herein relate to producing lighting effects for data in a volume data set with positions on a display, for use in rendering data from the data set on the **2**D graphics display **24** of the computer system **10**. The methods described herein modify lighting attributes already associated with the data in the volume data set, to produce lighting effects and pass the modified lighting attributes to other programs, or to a display buffer **316** in the RAM **18** for use by a display routine residing in the program memory **16**, for example, which uses information in the display buffer to activate pixels with appropriate intensity. This may involve storing Red (R), Green (G) and Blue (B) intensity values in association with a particular linear display position, for example.

Referring to Figure **2**, the methods described herein are intended to be used with a volume data set representing geometric and attribute information about voxels in a voxel block such as shown at **30**. Generally, a data acquisition device (not shown) such as a medical imaging device, x-ray device or computer axial tomographic (CAT) x-ray system, for example, may be used to generate geometric and attribute data for a plurality of voxels **32** in the voxel block **30**. Alternatively, the volume data set may be received from a graphics application program running on the processor circuit **12** shown in Figure **1** or may be provided to the processor circuit **12** from an external source through the media reader **19** or communications interface **21**, for example. The graphics application program may facilitate entry of volume data through the use of a keyboard or mouse, for example. In this embodiment, the volume data set provided by any source includes geometric and attribute information including visual information such as color, surface, normal vectors, density and layer information, for example. Attribute information is not limited to visual information or to the examples given here and could include other information, such as smell, sound or tactile information, for example. The type of information included as an attribute may also be user-defined.

Referring back to Figure **2**, in this embodiment the voxel block **30** for which the volume data set represents data is a generally orthorhombic shaped **3**-dimensional space divided into a regular grid defining a plurality of generally cubic sub-regions (voxels **32**) of the voxel block **30**.

Alternatively, the voxel block **30** may be any other solid-shaped **3**-dimensional space divided into a plurality of smaller sub-regions. For example, the voxel block **30** may be approximately cylindrically shaped and the voxels may be small cubic regions or small regions of solid angles formed in columns.

In this embodiment, the voxel block **30** is defined as an orthorhombic block as this shape is easily mapped into a **2**-dimensional coordinate system under which most computer displays operate. Other display or annunciation systems however, may be more amenable to use with differently shaped voxel blocks.

In this embodiment, the voxel block **30** is defined by an origin **33** and maximum x, y and z voxel coordinate positions **34**, **36** and **38** along orthogonal x, y and z axes **40**, **42** and **44** intersecting the origin **33**. The origin **33** and x, y and z axes together define a voxel coordinate system.

In this embodiment, a base **46** of the voxel block **30** lies in an x-z plane **48** established by the x and z axes while one side of the voxel block **30** is coincident with a y-z plane **50** established by the y-z axes and another side is coincident with an x-y plane established by the x-y axes. In this embodiment, a geometric position of each voxel **32** within the voxel block **30** can thus be represented by a three-tuple (x, y, z) specifying first, second and third

Cartesian coordinates x, y and z. Attributes associated with a voxel **32** are specified in connection with the geometric position. Thus, the properties of each voxel **32** can be specified by a voxel record comprised of a representation of the geometric position of the voxel in the voxel block **30** and a representation of attributes associated with the voxel. In this embodiment, voxel records of the volume data set may therefore be said to have the following form: Voxel:(x, y, z, a**1**, a**2**, a**3**, ... an), where x,y,z and a**1**,.. an are elements of the record and the x,y,z elements specify geometric coordinates of the voxel **32** in the voxel block **30** and the elements a**1**,...an specify n attributes associated with the voxel **32**. Alternatively, other coordinate systems may be used and converted to Cartesian coordinates for direct use with the methods disclosed herein.

In general, the voxel block **30** may have non-active voxels such as shown at **60** having no representative information, such as measured information below a threshold level, and may alternatively or further include active voxels such as shown at **62** for which information has been obtained or provided. Effectively, active voxels **62** may represent an object within the voxel block **30** and hence the volume data set may be considered to define an object model which can be specified by three components including: **1**) voxel block, or array parameters, i.e., the maximum x, y and z coordinate positions **34**, **36** and **38** of the voxel block (X_size, Y_size, Z_size); **2**) an attribute definition defining the attributes associated with each voxel **32**; and **3**) a plurality of active voxel records each comprised of geometric information and attributes. In this embodiment the geometric information is represented by a three-tuple (x,y,z), and the attributes are specified according to the attribute definition.

The voxel block **30** may be considered to have a plurality of columns on the x-z plane, one of which is shown at **52**, at the x-z position (**4**, **10**). A column of active voxels **53** may be defined as being comprised of only the group of active voxels within a column **52**. For example, the column **52** defined at the x-z position (**4**, **10**) has five active voxels at y-positions **1**,**2**,**5**,**7** and **10**, and has six non-active voxels at y-positions **0,3,4,6,8** and **9**. Thus the column of active voxels **53** is considered to be comprised only of the voxels at y-positions **1,2,5,7** and **10**. This avoids wasting memory locations that would otherwise be used to store information about inactive voxels.

The voxel associating method and apparatus described herein are intended for use with voxel information stored in a column-oriented data structure. A suitable column oriented data structure is described in International Patent Application No. PCT/CA**01/00686**, filed May **16, 2001**. Generally in a column oriented data structure of the type referred to herein, a column of active voxels in a voxel block is associated with a group of memory locations, each of which is associated with a respective index and at least the geometric representations of active voxels in the column are stored in successive memory locations in the group. This permits a group of memory locations associated with a column of active voxels in the voxel block to be addressed. The individual memory locations within the group can then be sequentially addressed, enabling fast access to voxel information, a column at a time, as will become apparent below.

An example of a column-oriented data structure with which the apparatus and methods according to the various embodiments of the invention operate is shown at **70** in Figure **3**. Broadly, this data structure includes a data area **72** and a management area **74**.

The data area **72** includes a plurality of memory locations, each of which is associated with an index, an example of which is shown at **76**. Any given index is associated with corresponding memory locations, or fields **78**, **80**, **82**, **84**, **86**, in respective arrays **90**, **92**, **94**, **96**, **98** that store information about a voxel, such as a y coordinate, a layer value, a normal value, a color value and a density value, respectively. Thus, once an index is known, all memory locations associated with that index are known and hence, all information relating to the corresponding voxel is known.

The management area **74** includes a macro variable storage area **100**, a base array **102** and a memory allocation control area **104**. Generally, the macro variable storage area **100** includes fields **110**, **112**, **114** for storing a three-tuple specifying the maximum x, y and z coordinates of the voxel block and includes fields **116**, **118**, **120** and **122** for storing data defining a function φ for converting points in a global coordinate system to points in the voxel coordinate system. Effectively, the macro variable storage area **100** is used to store variables, which are used in an initialization routine described below.

The memory allocation control area **104** is used to allocate memory to the data structure and is not relevant in associating. However, during formation of the data structure and during storage and editing of data therein, the memory allocation control area essentially associates indices such as shown at **76** with memory locations such as shown at **78**, **80**, **82**, **84** and **86**.

The base array **102** is a two dimensional array that associates groups of indices in the y-array **90**, and hence associates groups of memory locations in each array **90**, **92**, **94**, **96** and **98** with columns of active voxels in the voxel block shown in Figure **2**. To do this, the base array **102** has a pair of index and count fields **130** and **132** associated with each column in the voxel block. The index and count fields store values respectively representing a starting index of a group of memory locations associated with a column of active voxels and a count of the number of indices following the starting index that relate to memory locations associated with active voxels in the same column. Columns are identified by the x and z coordinates of the voxel, hence the base array is a two dimensional array that is addressed by the x and z coordinates of a column. Thus, given the x and z coordinates of a column, the index field gives the index of the first memory location associated with the column, and the count field gives the number of memory locations associated with the column.

Since the y coordinates of the active voxels in the column are stored in association with respective indices in the group identified by the base array, the x and z coordinates of a given voxel can be used to find the group of memory locations associated with the column and the y coordinate of the given voxel can be used to search within the group for the associated index. This index identifies all the memory locations in which all the attributes of the voxel are stored, for use in creating lighting effects.

Of particular interest to creating lighting effects are the normal and color arrays **94** and **96** shown in Figure **3**.

Normal Array

The normal array is used to store representations of vectors normal to respective voxels in the voxel block. Thus each entry in the normal array holds a representation of a normal vector normal to the surface of the object represented by the voxel block at the voxel associated with the entry. The normal vectors are preferably pre-calculated by the source of the data stored in the data structure, but if not already provided may be calculated using the usual methods for calculating same, such as by calculating a perpendicular to the gradient of the surface at the voxel. Such calculations may produce a large number of normal vectors and therefore to ensure speed in creating lighting effects it is desirable to quantize the directions of the normal vectors to a finite number of directions.

To quantize the possible directions of normal vectors, a unit hemisphere centered on the origin and extending thereabove in the y direction is divided into a plurality of solid angle regions in **64** horizontal azimuth and **64** vertical zenith directions. Thus there are **2^{12}** (**4096**) possible normal directions. Similar unit hemispheres centered on the origin and extending in the x direction and in the z direction are also each divided into **4096** possible normal directions.

The calculated normal is compared to the quantized directions of the three unit hemispheres to find the nearest quantized value in any hemisphere. This nearest quantized value acts as a compressed normal value and is stored along with an identification of the hemisphere, in the normal field for the associated voxel. Thus the normal fields contain compressed normal values from **0** to **4095** and direction values from **1** to **3**, representing one of the quantized normal directions in one of the three hemispheres.

The normal tables could just as easily be represented by a single table with **8192** entries, for example, for holding quantization values for two hemispheres with coterminous equators, that is, a sphere.

By quantizing the normal directions, the time taken to create lighting effects is reduced, as will become apparent below. Larger quantization areas will produce a smaller number of quantized directions and thus reduce the memory requirements, at the expense of loss of resolution in lighting effects. The number of quantized directions described herein has been found to enable the production of lighting effects of a resolution which is generally not detectable by the human eye, when viewed on typical computer displays.

Color Array

In this embodiment, each entry in the color array **96** may hold a word representing a single intensity value Io representing an intensity value for lighting a pixel on the display **24** shown in Figure **1**, when the corresponding voxel is mapped to the display. A single value Io would be used, for example, in a black and white rendering, where varying intensities Io provide lighting intensity values in a gray scale from complete black to white. If the intensity values are represented by bytes of data in the color array, **256** shades of gray ranging from complete black to white could be represented.

Alternatively, each entry in the color array **96** may be comprised of fields for storing color words which may be comprised of any number of bytes or bits, representing the intensity that is to be applied to corresponding red, green or blue controls for creating colored light at the pixel to which the related voxel is mapped. In this embodiment, the color word is comprised of two bytes in which **5**, **6** and **5** bits represent Red, Green and Blue intensities respectively. The use of a larger number of bits for the Green intensity provides greater resolution in the green portion of the spectrum, which is in accordance with retinal sensitivity of human observers.

Referring back to Figure **1** the processor circuit-readable codes which program the processor circuit **12** cause the processor circuit to interact with the data structure **70** shown in Figure **3** to modify lighting attributes such as color, already associated with the data in the volume data set, to produce lighting effects and to pass the modified lighting attributes to other programs or to the display buffer **316** for use by the display routine **17** which uses information in the display buffer to activate pixels on the display **24** with appropriate intensity.

Referring to Figure **4** to produce lighting effects, the voxel block is considered to exist in a global coordinate system **134**. An imaginary viewing plane **135** is specified by an orientation and position D in the system and defines the direction of a viewing vector V on a line extending between the center voxel C in the voxel block **30** and the position D.

At least one imaginary lighting plane **136** comprised of a plurality of lighting elements **137** at respective lighting plane positions is specified by a location P in the system **134**. The location P may be considered to be an origin in a lighting coordinate system. A line extending between the center voxel C in the voxel block and the position P in the lighting plane defines the direction of an illumination vector L. The imaginary lighting plane **136** is considered to be of a length and width sufficient to cast imaginary light from the lighting elements in respective parallel paths to illuminate certain voxels of the voxel block, which map into the lighting elements in the lighting plane. In this embodiment, the length and width are two times the maximum X_size, Y_size or Z_size of the voxel block **30**.

Lines parallel to the viewing vector V emanating from voxels in the voxel block **30** intersect the viewing plane **135** and would define an outline **138** of the voxel block **30** if viewed from the direction of the viewing plane.

Referring to Figure **5**, a flowchart representing the overall process of creating lighting effects as implemented by the processor-readable codes is shown generally at **140**. The process involves two routines including an initialization routine **142** and a lighting routine **144**. The initialization routine is executed using data from the macro variable storage area **100** and the known position P of the lighting plane **136**. More than one imaginary lighting plane may be considered, in which case the initialization routine **142** may have to be executed once for each imaginary lighting plane, such as shown in Figure **6**.

In this specific embodiment, referring to Figures **7**A and **7**B, the initialization routine **142** involves computing a projection map, computing a scan order, computing address conversion tables, computing a half-way vector, computing specular and diffuse illumination tables, and initializing and loading a priority buffer with voxel references.

Initialization Routine

Effectively the initialization routine involves computing specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel and associating the specular and diffuse illumination values with respective normal vectors. This facilitates retrieval of the specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced.

In this embodiment, the identification of voxels upon which lighting effects are to be produced are also determined by the initalization routine by scanning voxel positions in the voxel block **30**. Scanning is performed according to a scan order in which column positions in respective planes through the voxel block **30** which are progressively farther from the lighting plane are addressed in order of those which map to foreground positions in the lighting plane before those which map to more background positions in the lighting plane. Scanning is performed to produce voxel position specifiers identifying illuminable voxel positions in the voxel block **30**. Then, respective lighting plane positions are determined for each position specifier according to a lighting plane position mapping. Respective lighting plane positions are then associated with corresponding position specifiers such that position specifiers which map to lighting plane positions which are already associated with position specifiers identifying more foreground positions are not associated with lighting plane positions. Thus voxels behind voxels in more foreground planes are occluded and only voxels that are visible to the light source are produced are identified.

The lighting routine is run after execution of a corresponding initialization routine and executes on receipt of a display position/voxel reference from another program (not shown) that associates display positions with voxel references. The lighting routine uses the received voxel reference to address the base array **102** to determine the corresponding voxel position in the base array, which is then mapped to a lighting plane position. The position specifier already associated with that lighting plane position by the initialization routine is compared to the determined voxel position. If the position specifier is the same as the determined voxel position, the corresponding display position in a display buffer is loaded with a lighting value produced by modifying the intensity or color values associated with the determined voxel position by the color array **96** to incorporate specular and diffuse illumination components. In this manner, only voxel references that identify voxel positions that have already been determined to be illuminated will have their lighting values modified with specular and diffuse illumination components for use at their associated display positions.

In the embodiment shown, referring to Figure **7**A, the first step in the initialization routine **142** is represented by block **150** and involves determining a projection map that maps points in the voxel coordinate system into points in the lighting plane coordinate system. A variety of different maps may be employed, depending on whether the imaginary light is considered to be a wide aperture flood source or a point source and the distance between the imaginary light and the voxel block in the global coordinate system. With a wide aperture flood source or a distant point source a parallel projection map may be used and with a point source near the voxel block a perspective projection map may be used. In this embodiment a parallel projection map is used as this simplifies the calculations, requiring only the position of the lighting plane **136** to be specified.

To determine a parallel projection map, it will be appreciated that the voxel block coordinate system (and hence the voxel block's position and orientation) may be described by a function φ that produces the voxel block coordinates of a point, given its global coordinates. The function φ employs a unique three-tuple $\mathbf{p}_\varphi$ and a unique **3**x**3** orthogonal matrix $\mathbf{R}_\varphi$ such that for every point $\mathbf{x}$ (given in global coordinates), the corresponding point in voxel block coordinates is given by:

$$\varphi(\mathbf{x}) = \mathbf{R}_\varphi\,\mathbf{x} + \mathbf{p}_\varphi$$

In addition to $\mathbf{R}_\varphi$ and $\mathbf{p}_\varphi$, which define the position and orientation of the voxel block, the three integers X_size, Y_size and Z_size are supplied to define the dimensions of the voxel block. $\mathbf{R}_\varphi$, $\mathbf{p}_\varphi$, X_size, Y_size and Z_size are all specified in the macro variable storage area **100** of the data structure **70** shown in Figure **3**.

The lighting plane coordinate system (and hence the view plane's position and orientation) may be described by a function ψ that produces the lighting plane coordinates of a point, given its global coordinates. The function ψ employs a unique three-tuple $\mathbf{p}_\psi$ and a unique **3**x**3** orthogonal matrix $\mathbf{R}_\psi$ such that for every point **x** (given in global coordinates), the corresponding point in lighting plane coordinates is given by:

$$\psi(\mathbf{x}) = \begin{bmatrix} 100 \\ 010 \end{bmatrix} (\mathbf{R}_\psi\, \mathbf{x} + \mathbf{p}_\psi)$$

In addition to $\mathbf{R}_\psi$ and $\mathbf{p}_\psi$, which define the position and orientation of the lighting plane, two integers X_plane_size and Y_plane_size may be supplied to define the dimensions of the lighting plane. In this embodiment these integers are the same and are two times the largest of X_size, Y_size and Z_size.

A parallel projection map π for a point **x** in the global coordinate system is defined by:

$$\pi(\mathbf{x}) = \mathbf{x} - ((\mathbf{x} - \mathrm{p}_\psi) \bullet \mathbf{n})\mathbf{n},$$

where

$$\mathbf{n} = \mathbf{R}_\psi \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus, a parallel projection map Π that produces the lighting plane coordinates of a point given its voxel block coordinates $\mathbf{x}_V$ is given by:

$$\Pi(\mathbf{x}_V) = (\psi \circ \pi \circ \varphi^{-1})(\mathbf{x}_V)$$

This general transform will be referred to as a parallel projection transformation and hereinafter referred to as Π. Thus in general, the transformation Π determines lighting plane coordinates (Xl, Yl) associated with voxel coordinates (Xv, Yv, Zv). Given any voxel coordinates (Xv, Yv, Zv) a corresponding lighting plane coordinate (Xl, Yl) can be produced.

The next step in the initialization routine **142** is represented by block **152**, which directs the processor circuit to compute a scan order. Effectively, the scan order is represented by a base point and plane and column axes in the voxel co-ordinate system. The base point is the scan starting point in the voxel block **30** co-ordinate system, and the plane and column axes specify the direction of incrementally addressing respective voxel positions in the voxel block **30** during the scan.

As a first step in finding the base point, an illumination orientation vector **u** is determined. Referring to Figure **8**, effectively this involves assigning two orthogonal vectors **164** and **166** or their corresponding parity reverse counterparts **168** and **170**, to define illumination plane axes in the lighting plane coordinate system. The illumination orientation vector **u** is the one of the vectors that is closest to a vector **162** in the lighting plane coordinate system that represents a mapping of the y-axis of the voxel block **30**. In this embodiment that vector is vector **166**. Mathematically, the determination of the illumination orientation vector can be represented as:

Let $\mathbf{u} \in \{(\mathbf{1},\mathbf{0}),(\mathbf{-1},\mathbf{0}),(\mathbf{0},\mathbf{1}),(\mathbf{0},\mathbf{-1})\}$ that maximizes $\mathbf{u} \bullet \Pi\begin{bmatrix}0\\1\\0\end{bmatrix}$

If more than one of these unit vectors maximizes the above statement, any one may be selected.

Next, referring to Figure **9**, determining the base point involves finding a corner vector, from the origin in the lighting plane coordinate system to a mapped corner point of the voxel block **30**, that produces the smallest dot product with the illumination plane orientation vector **u** (**162**).

To do this, each of the eight voxel positions **181**, **182**, **183**, **184**, **185**, **186**, **187**, **188** at the corners of the voxel block **30** is mapped, using the parallel projection transform, into corresponding positions **191**, **192**, **193**, **194**, **195**, **196**, **197**, **198** in the lighting plane coordinate system. Each of these corresponding positions may be considered to define respective **corner vectors**, one of which is shown at **200** in Figure **9**, extending from the origin in the lighting plane coordinate system.

The base point is defined as the corner point of the voxel block **30**, in the voxel coordinate system, associated with the corner vector in the lighting plane coordinate system, that when normalized produces the smallest dot product with the illumination orientation vector **u** (**166**). To find this point, the scalar values representing the dot products of each of the eight normalized corner vectors with the illumination orientation vector **u** (**166)** are found. The base point is the corner point associated with the normalized corner vector **200** which forms the largest angle with the illumination orientation vector **u**. In the example shown in Figure **9**, the base point may be point **184**. Mathematically, the determination of the base point may be represented as :

Let **b** be a corner of the voxel block (ie **b** $\in$ {(**0**, **0**, **0**), (**0**, **0**, Z_size), (**0**, Y_size, **0**), (X_size, **0**, **0**), (X_size, Y_size, **0**), (X_size, **0**, Z_size), (**0**, Y_size, Z_size), (X_size, Y_size, Y_size)} that minimizes $\mathbf{u} \bullet \Pi\left(\frac{\mathbf{b}}{\|\mathbf{b}\|}\right)$.

After finding the base point, it is necessary to determine the plane and column axes of the voxel block **30**. Determining the plane axis involves determining an edge of the voxel block **30** that contains the base point **184**, and the projection of which forms the smallest angle with the illumination orientation vector **u**. The column axis is the edge of the voxel block that contains the base point **184** the projection of which forms the greatest angle with the illumination orientation vector **u**.

To do this, referring to Figure **10**, two unit vectors **202** and **204** extending from the base point **184** along the x and z axes respectively of the voxel block are mapped using the Π transform into corresponding **x edge** and **z edge** vectors **212** and **214** in the lighting plane coordinate system. An angle is formed between each of the **x edge** and **z edge** vectors **212** and **214** and the illumination orientation vector **u 166**, and the edge associated with the smallest angle is considered to be associated with the plane axis while the other is considered to be associated with the column axis. Thus, the plane axis will be either the x or z axis **40**, **44** of the voxel block **30** and the column axis will be the other. In the example shown, the selection of the lighting plane has resulted in the same dot product for each edge vector. Thus, either one can be the plane axis. Take the z-axis **44** as the plane axis, for example. The x-axis **40** will be the column axis.

Mathematically, the determination of the plane and column axes may be represented as:

Let **r** be the unique corner of the voxel block such that **r-b** is parallel to the x-axis of the voxel block coordinate frame.

Let **s** be the unique corner of the voxel block such that **s-b** is parallel to the z-axis of the voxel block coordinate frame.

Then, if $\Pi\left(\frac{\mathbf{r}-\mathbf{b}}{\|\mathbf{r}-\mathbf{b}\|}\right)\bullet\mathbf{u} > \Pi\left(\frac{\mathbf{s}-\mathbf{b}}{\|\mathbf{s}-\mathbf{b}\|}\right)\bullet\mathbf{u}$, the plane axis is the z axis, otherwise the plane axis is the x axis.

Referring back to Figure **7A**, next the processor circuit is directed to block **154**, which causes the processor circuit to compute address conversion tables for mapping voxel positions in the voxel block **30** into corresponding lighting plane positions. These lighting plane positions may be linear addresses identifying individual lighting elements **137** on the lighting plane **136**, for example. In this embodiment, three address conversion tables are produced as shown in Figure **11**. These are identified as plane, column and y-direction tables. The plane table holds values representing integer multiples of a linear address transform of a point in the lighting plane coordinate system that represents a point representing a unit vector on the plane axis, in this example along the z-axis **44** in a negative direction from the base point in the voxel block coordinate system. The plane transform may thus be represented as $Vp(\Pi(0,0,1))$ where $(0,0,1)$ represents the unit vector on the plane axis, Π is the parallel projection transform to transform the unit vector into lighting plane coordinates and Vp represents a contribution to the lighting plane address in the plane direction. Similarly, the column table holds values representing integer multiples of a linear address transform of a point in the lighting plane coordinate system which represents a point representing a unit vector on the column axis. The column transform may thus be represented as $Vc(\Pi(1,0,0))$. The y-direction table holds values representing integer multiples of a linear address transform of a point in the lighting plane coordinate system which represents a point representing a unit vector parallel to the y axis **42** in the voxel block coordinate system and extending away from the base point. The y transform may thus be represented as $Vy(\Pi(0,1,0))$.

The full transform for obtaining a linear lighting plane address from a three-tuple (p,c,y) representing a voxel position in the voxel block is given as:

$$A_d=V_o+pVp(\Pi(0,0,1))+cVc(\Pi(1,0,0))+yVy(\Pi(0,1,0))$$

Since each position p,c,y in the voxel block **30** can effectively be represented as integer extensions of unit vectors in each direction from the base point, the above function is easily implemented in a fast lookup table as shown in Figure **11**. The p,c,y coordinates are used as indices into the lookup table. The value A_d produced by the transform in this embodiment represents a linear address of the lighting plane or more generally a lighting element position.

Alternatively, other transforms may be used to produce lighting element positions wherein voxel positions are mapped in different ways. For example, a shear transformation followed by a warp transformation may be incorporated into the Vp, Vc, Vy transforms to effectively map voxel positions to lighting element positions in a manner that arranges planes of voxels into a priority order.

In the event that the x-axis had been found to be the plane axis, the entries under the Plane (z) and Column (x) headings would be interchanged and the headings would be changed to Plane (x) and Column (z).

Next, an optional block **155** directs the processor circuit to recognize clipping tables. These tables are optional and may be provided from an outside source. If used, these tables may be recognized by providing the processor circuit with an address identifying where in memory they can be found. Effectively a clipping table includes fields associated with corresponding columns in the voxel block **30** and these fields are populated with indicators identifying columns in the voxel block for which position specifiers are to be produced and/or are populated with indicators specifying a range of y-values within the associated column for which position specifiers are to be produced.

A clipping table may have the same dimensions as the base array shown in Figure **3**, for example. In one implementation, there may be two clipping tables, one for column inclusion and the other for range definition. One or the other or a combination of both may be employed. An example of a column inclusion-clipping table is shown at **221** in Figure **12**. Note that a number of fields in rows **220** are populated with ones while the remaining fields in rows

222 are populated with zeros. This has the effect of producing a clipping plane **224** in the voxel block, as shown in Figure **13**, such that voxels in the shaded area **226** are not included in the render. Generally boundaries defined by ones and zeros in adjacent rows define clipping planes parallel to the y-z plane of the voxel block while boundaries defined by ones and zeros in adjacent columns define clipping planes parallel to the x-y plane of the voxel block. It will be appreciated that by populating the clipping table **221** with any arrangement of ones and zeros clipping boundaries of any convex shape can be defined. Effectively, the column inclusion-clipping table **221** shown in Figure **12** may be used to define which columns are to be included in the render and which are not.

Similarly, a range clipping table as shown at **223** in Figure **12** may define limits in the y-direction between which voxels are to be included in the render and outside of which voxels are not to be included in the render. In this table **223**, each position in the table identifies the range of y values such as **0** to **50**, for example, which are to be included in the render. This has the effect of defining clipping boundaries vertically within the voxel block **30**, to create clipping planes parallel to the x-z axes of the voxel block as shown in Figure **14**. By populating the range clipping table appropriately, clipping boundaries of any convex shape can be defined. The optional interaction with the clipping tables will be described in connection with the lighting routine shown in Figure **16**.

Referring back to Figure 7A, the next step in the initialization routine is represented by block **156** which directs the processor circuit to compute a half-way vector **H** defined as a unit vector which is half-way (in angle) between the unit illumination vector **L** representing a direction to the lighting plane **136** and the normalized viewing vector **V** representing a direction to a viewing plane **135** in the global coordinate system shown in Figure **4**. Thus only the direction of the unit illumination vector **L** and the direction of the normalized viewing vector **V** are required to compute the half-way vector **H**.

Referring back to Figure **7**A, the next step in the initialization routine is represented by block **157** which directs the processor circuit to compute specular and diffuse illumination tables as shown at **161** and **163** respectively in Figure **15**. The specular illumination tables associate specular illumination values with the compressed normal values stored in the data structure **70**. Specular illumination values **165** are calculated as the dot product of the unit normal vector **N,** associated with the compressed normal value, with the half-way vector **H** raised to a power **k** indicative of the "shininess" of the surface:

$$S = (\mathbf{N} \bullet \mathbf{H})^k$$

Since the range of normal vectors is known to be **0** to **4095** for each of three hemispherical spaces, all possible specular illumination values S in hemispherical space can be pre-calculated and stored in a table having separate portions for each hemispherical space. The specular illumination table contains all of these values, indexed by compressed normal values **167** where the direction values are used to index to the respective portion in the table associated with the corresponding hemispherical space. Thus, given a compressed normal value **167**, a corresponding specular illumination value **165** can be found from the specular illumination table **161**.

Similarly, the diffuse illumination tables **163** associate diffuse illumination values **169** with the compressed normal values **167** stored in the data structure **70**. Diffuse illumination values **169** are calculated as the dot product of the unit normal vector **N**, associated with the compressed normal value, with the normalized illumination vector **L**:

$$D = \mathbf{N} \bullet \mathbf{L}$$

Since the range of normal vectors is known to be **0** to **4095** for each of the three hemispherical spaces, all possible diffuse illumination values S in hemispherical space can be pre-calculated and stored in a table having separate portions for each hemispherical space. The diffuse illumination tables contain all of these values, indexed by compressed normal value **167** where the direction values are used to index to the respective portion of the table associated with the corresponding hemispherical space. If the dot product of normalized V and L is negative, that is, the angle between them is greater than **90** degrees, then the vector L is negated and a new halfway vector H is calculated and these new L and H vectors are used in producing the diffuse and specular tables. Thus, given a compressed normal value **167**, a corresponding diffuse illumination value **169** can be found from the appropriate diffuse illumination table **163**.

Referring back to Figure **7A**, block **158** of the initialization routine involves the initialization of a priority buffer shown at **225** in Figure **1** for storing the priority table. The priority buffer **225** is formed in the memory **18** and may be considered to be a linear array of the same size as the linear address range of the lighting plane **136**, for holding voxel references identifying illuminable voxel positions in the voxel block **30**. Initially, all values in the priority buffer **225** are initialized to a predefined null value, and the buffer positions are populated with voxel references by a priority routine portion **143** as shown in Figure **7**B, of the initialization routine **142**, to indicate voxel positions that are illuminated.

Priority Routine

Referring to Figure **7B**, the priority routine **143** includes a voxel addressing and verification portion **230** and a reference assignment portion **232**. The voxel associating and verification portion **230** effectively causes the processor circuit to scan the voxel block **30** using the scan order determined at block **152** of the initialization routine to produce position specifiers identifying potentially renderable voxel positions in the voxel block **30**. This involves producing voxel addresses according to the scan order and determining whether a voxel at each address produced is active and/or clipped before being passed to the reference assignment portion **232**. The reference assignment portion **232** causes the processor circuit to compute a lighting plane position from the position specifier produced by the voxel addressing and verification portion, using the address conversion tables produced at block **154** of the initialization routine, and to associate a reference to the voxel identified by the position specifier with the computed lighting plane position if no reference has already been associated with that position.

The voxel addressing and verification portion **230** involves a first block **234** which causes the processor circuit to produce column addresses beginning at the base point **184** and advancing along the column axis for each position along the plane axis. Initially, as shown in Figure **12**, a plane axis position of zero is selected, defining a first, closest plane **235** on the zero position of the plane axis and extending in the y-direction of the voxel block **30**, closest to the illumination plane. This first plane contains the base point **184**. Next, a column axis position of zero, relative to the base point **184** is selected, defining a first column **237** address of a column of voxels containing the base point. This column **237** is closest to the lighting plane.

Referring back to Figure **7**B, after a column address has been determined, blocks **236** and **238** direct the processor circuit to identify whether the column addressed is within a renderable column range, as part of determining whether a voxel is renderable. Block **236** directs the processor circuit to determine whether the column at that address contains active voxels. This is done by converting the determined column address into x-z coordinates in the voxel block coordinate system and addressing the base table shown in Figures **3** and **12** with these x-z coordinates, to determine whether the index field **130** contains a valid index value. If it does not, there is no starting index of active voxels in the column and thus no active voxels in the column. If this is the case, the processor circuit is directed back to block **234** of Figure **7**B to set a new column address identifying the next column in the first plane, which is the next most foreground column in the plane. If addresses for all columns in the first plane have been produced, the plane axis coordinate is advanced to identify a next closest plane behind the first plane and the column axis coordinate is sequenced through coordinates from closest to farthest. This has the effect of addressing planes in a succession of planes which are progressively farther from the base point and producing column addresses of columns progressively farther from an edge of the voxel block, in the addressed plane.

If at block **236** the index field **130** is found to contain a valid index, the column addressed by the current column address is identified as having active voxels. The processor circuit may then optionally be directed to block **238** which causes it to use the converted column address, in x-z coordinates of the voxel coordinate system, to address the column inclusion clipping table **221** shown in Figure **12.** The x-z coordinates in the voxel coordinate system may then be used to locate the corresponding position in the clipping table to determine whether it contains a one or a zero. If it contains a zero, the column has been clipped and is not renderable and the processor circuit is directed back to block **234** to set a new column address.

If the clipping table **221** position contains a one, the column is renderable and the processor circuit is directed to block **240**, which directs it to set a voxel address within the column. To do this, the processor circuit is directed to the starting index of the first active voxel in the column, as determined by the contents of the index field **130** in the base array position identified by the x-z coordinates of the column address, as shown in Figure **12**. The processor circuit is then directed to use the starting index to address the y-array **90** of the data structure **70** shown in Figure **3** to locate a y-coordinate associated with the starting index.

Next, referring back to Figure 7B, optionally, the processor circuit may be directed to block **242**, which causes it to compare the y-coordinate to the range of values specified in the associated column position in the range clipping table shown at **223** in Figure **12**, specified by the x-z coordinates in the voxel coordinate system. If the y-coordinate is not within the range specified, the processor circuit is directed back to block **240** of Figure **7**B which causes it to find the y-coordinate of the next index in the active column specified by the column address. Thus, the processor circuit loops through blocks **240** and **242** until there are no more indices associated with active voxels in the addressed column, at which time it is directed back to block **234** to determine a new column address. Or, if during the above looping, a y-coordinate within the range specified by the indicated position in the range clipping table is found, the column address and the y-coordinate together are considered to be a position specifier specifying a renderable voxel position in the voxel block. The processor circuit is then directed to the reference assignment portion **232** of the priority routine **143**.

As a result of completing the voxel addressing and verification routine the processor circuit has effectively produced a renderable voxel position specifier in the form of a three-tuple comprised of a plane axis value, a column axis value and a y-value. This three-tuple is received at a first block **250** of the reference assignment portion, which directs the processor circuit to compute a lighting plane position based on the three-tuple produced by the voxel addressing and verification portion **230**, using the lighting plane position mapping provided by the lookup tables shown in Figure **11**.

Once the lighting plane position is known, it is used at block **252** in Figure **7**B to address the priority buffer **225** to determine whether the corresponding position has already been loaded with a non-null value. If such position contains a non-null value, that position has already been associated with a voxel reference and therefore no action is required. The processor circuit is then directed back to block **240** to advance to the next voxel position in the addressed column.

If at block **252** the corresponding position of the priority buffer **225** contains a null, no reference has yet been associated with that lighting plane position and the processor circuit is directed to block **255** which causes it to associate a reference, in this embodiment the index **76** associated with the voxel position identified by the voxel position specifier, with the lighting plane position produced at block **250**.

The processor circuit is then directed back to block **240** to advance to the next voxel position in the addressed column and to produce a new position specifier.

As a result of the scan order, during execution of the priority routine the closest plane having renderable voxels is considered first, the next closest plane is considered and so on, until the farthest plane from the light source is considered. This ensures that references to renderable voxels in the closest planes are associated with lighting plane positions before references to renderable voxels in the farthest planes. Consequently, the nearest planes are given priority in assignment of references to lighting plane positions, which automatically occludes renderable voxels in farther planes that would map to lighting plane positions with which references have already been associated.

When the entire voxel block **30** has been scanned and references have been assigned, the initialization routine **142** is considered to be completed, leaving the address conversion tables shown in Figure **11** and the priority buffer **225** shown in Figure **1** available for use by the lighting routine **144** shown in Figure **16**.

Referring to Figure **16**, the lighting routine **144** waits for a display position/voxel reference to be received from an associating program (not shown) that associates display positions on the viewing plane with references to voxels. In this embodiment, these references are the indexes such as shown at **76** in Figure **3**.

Using the reference portion of the received display position/reference pair as an index, a first block **300** of the lighting routine **144** directs the processor circuit to fetch color, normal, and x,y,z voxel position values from the data structure **70** shown in Figure **3**.

Then block **302** directs the processor circuit to convert the x,y,z voxel position into plane, column and y coordinates (p,c,y) and to use these coordinates to address the address conversion table shown in Figure **11** to find the corresponding lighting plane position. The reference associated with this lighting plane position is then read from the priority buffer **225** and compared for equality with the received voxel reference that initiated the lighting routine **144**. If the references do not match, the voxel identified by the received reference is not illuminated due to its position in the voxel block relative to the lighting plane. In this case, block **304** directs the processor circuit to set diffuse and specular lighting values D and S to zero, for use in later calculations.

If at block **302** the references do match, the voxel identified by the received reference is illuminated. In this case the processor circuit is directed to block **306** which causes it to use the compressed normal value from the normal array **94** in Figure **3** to find the corresponding diffuse illumination value **169** in the diffuse illumination table **163** of Figure **16** and to assign it to the diffuse illumination variable D. Block **308** then directs the processor circuit to use the compressed normal to find the corresponding specular illumination value **165** in the specular illumination table **161** and assign it to the specular illumination variable S.

Then, block **310** directs the processor circuit to calculate an intensity value I, according to the relation:

$$I = A + D + S$$

Where:	I is an intensity value which is used as a multiplier for adjusting the intensity or degree to which the pixel at the corresponding display position is to be lighted;
	A is an ambient light value representing an ambient light on the voxel;
	D is the diffuse illumination value **169** representing the amount of diffuse light on the voxel
	S is the specular illumination value **165** representing the amount of specular light on the voxel

Block **312** then directs the processor circuit to use the intensity value I as a multiplier to increase or decrease the intensity specified by the RGB components RED_{in}, GRN_{in}, $BLUE_{in}$ of the word fetched from the color array **96** in Figure **3** to produce RGB output intensity values, RED_{out}, GRN_{out}, $BLUE_{out}$ for use by a display routine to drive RGB drivers of the display **24** shown in Figure **1**.

This is shown in more detail in Figure **17** in which a first block **320** directs the processor to extract the five most significant bits from the **16**-bit color word. These form the Red_{in} value. Next block **322** directs the processor to multiply the Red_{in} with the intensity value calculated at block **310** of Figure **16** and then block **324** directs the processor to send the eight most significant bits of the product to a RED_{out} location associated with the display position in the display buffer **316** in the RAM shown in Figure **1**, for use by the display routine.

Still referring to Figure **17**, the processor is then directed to block **326** of the output routine which causes it to extract the next six most significant bits from the color word; these form $Green_{in}$ value. Block **328** then directs the processor to multiply the Green_{in} value by the intensity value calculated at block **310**. Block **330** then directs the processor to send the eight most significant bits of the product to a GRN_{out} location in the display buffer **316** in the RAM shown in Figure **1** for use by the display routine. Finally, block **332** directs the processor to extract the next five most significant bits of the color word, i.e., the five least significant bits of the color word; these form Blue_{in} value. Block **334** then directs the processor circuit to multiply the Blue_{in} value by the intensity value produced at block **310**. Block **336** then directs the processor circuit to send the eight most significant bits of the product to a BLUE_{out} location associated with the display position in the display buffer **316** in the RAM shown in Figure **1** for use by the display routine.

While the methods and apparatus described above have been described as being implemented by processor readable codes and a processor circuit, various aspects of the apparatus may be implemented in hardware devices or a single hardware device such as an application specific integrated circuit (ASIC). In this regard, an apparatus according to an alternative embodiment of the invention is shown generally at **400** in Figure **18**.

The apparatus **400** includes a processor circuit **402** and memory **404**, which may include the processor circuit **12** and memory **18** shown in Figure **1**. In this embodiment, the processor circuit **402** is programmed to compute specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel of a voxel block and to store such values in the memory **404** to associate the specular and diffuse illumination values with respective vectors to facilitate retrieval of theses values when presented with a unit normal vector.

In this embodiment the apparatus further includes an application specific integrated circuit ASIC (**406**) that performs many of the functions described in connection with the first embodiment. In particular, the ASIC includes an identifier **408** in communication with the processor and with the data structure **70**, which in this embodiment is shown as being in the memory **404**, but may alternatively be in another memory. The ASIC further includes a modifier **410** in communication with the memory **404** to access the specular and diffuse illumination values and in communication with a display buffer **412**, which may be the display buffer shown in Figure **1** or another display buffer. The ASIC further includes memory **414**, which acts as a priority buffer, and includes a mapper **416**, which maps voxel positions in the voxel block into lighting plane positions. Finally, the ASIC includes a receiver **418** operable to receive display position/voxel reference pairs from another device, which may be the processor **402** or another processor operable to associate display positions with voxel references. The receiver **418** is in communication with the modifier and provides data to the modifier for it to operate on.

Effectively, the processor circuit **402** receives V view plane and L lighting plane directional vectors from another program running on itself or from another processor and then proceeds to associate specular and diffuse illumination values with all of the unit normal vectors in a set of unit normal vectors pointing in respective directions quantized to a finite number of directions, as described in connection with the first embodiment. The specular and diffuse values are associated with the unit normal vectors by tables as shown in Figure **15**. These tables are stored in the memory **404** and thus are accessible by the modifier **410**.

The identifier **408** includes a hardware implementation of a scanner **420** that scans the voxel block as described in connection with the first embodiment. Procedures for determining the base point and scan order are pre-configured to automatically calculate these parameters given access to the data structure **70** and given access to the view plane direction and lighting plane direction provided by the processor circuit **402**. In addition, the mapper is pre-configured to automatically calculate and store address conversion tables of the type shown in Figure **11** for use in mapping voxel block positions into lighting plane positions. In effect, the functions of the initialization routine shown in Figure **7**A are automatically carried out by the functional blocks of the ASIC shown in Figure **18**.

Then the scanner **420** automatically executes the priority routine shown in Figure **7**B to scan the voxel block to produce illuminable voxel identifiers and to load the priority buffer **414** with references to voxels that are illuminated. Thus, the specular and diffuse illumination tables are ready, the priority buffer is ready, and the mapper is ready for use by the modifier.

The modifier receives a display/voxel reference from the external source and communicates with the mapper to find the lighting plane position associated with the given voxel reference. Then, the corresponding lighting plane position in the priority buffer is addressed to find the associated voxel reference. If the found voxel reference matches the received voxel reference, an intensity value is calculated by using the unit normal vector stored in the data structure in association with the voxel reference, to index the specular and diffuse illumination tables in the memory **404** to retrieve the corresponding specular and diffuse illumination values for use in the equation shown in block **310** in Figure **16**.

The modifier then uses the calculated intensity value to modify the RBG color intensity values stored in association with the received voxel reference to produce the RED_{out}, GRN_{out} and $BLUE_{out}$ values to be stored in the display buffer **412**.

While specific embodiments of the invention have been described and illustrated, such embodiments should be considered illustrative of the invention only and not as limiting the invention as construed in accordance with the accompanying claims.

What is claimed is:

1. A method of producing a lighting effect on voxel data, the method comprising:

    computing specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel; and

    associating said specular and diffuse illumination values with respective said vectors to facilitate retrieval of said specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced.

2. The method of claim **1** wherein computing specular and diffuse illumination values comprises producing a table.

3. The method of claim **1** wherein computing specular illumination values comprises computing the direction of a unit half-way vector H at an angle half-way between an illumination vector L representing a direction to a lighting plane and a viewing vector V representing a direction to a view plane.

4. The method of claim **3** wherein computing specular illumination values comprises producing a table associating a plurality of discrete unit normal vectors with corresponding specular illumination values calculated according to the relation:

    $S = (N \bullet H)^k$

    WhereS is the calculated specular illumination value

    N is a discrete unit normal vector from said set of normal vectors

H is a unit vector half-way between a unit view vector V and a unit lighting vector L

k is an exponent representing a property of a surface

5. The method of claim **4** wherein producing diffuse illumination values comprises producing a table associating a plurality of discrete normal vectors with corresponding diffuse illumination values calculated according to the relation:

$$D = N \bullet L$$

WhereD is the calculated specular illumination value

N is a discrete unit normal vector from said set of normal vectors

L is a unit illumination vector

6. The method of claim **5** wherein producing diffuse and specular illumination values comprises negating L when the dot product of normalized V and L vectors is negative.

7. The method of claim **6** wherein producing said specular and diffuse illumination tables comprises apportioning said tables into portions associated with respective areas of a sphere within which said normal vectors may lie.

8. The method of claim **1** further comprising identifying a voxel upon which lighting effects are to be produced.

9. The method of claim **8** further comprising modifying prestored lighting values associated with an identified voxel according to at least one of said specular and diffuse illumination values determined using a normal vector of said identified voxel.

10. The method of claim **9** wherein identifying comprises scanning said voxel block.

11. The method of claim **10** wherein scanning said voxel block comprises scanning according to a scan order in which column positions in respective planes through the voxel block that are progressively farther from a lighting plane are addressed in order of those that map to foreground positions in the lighting plane before those which map to more background positions in the lighting plane.

12. The method of claim **9** wherein identifying comprises identifying potentially illuminable voxels in said voxel block.

13. The method of claim **12** wherein identifying potentially illuminable positions comprises scanning said voxel block to produce position specifiers identifying said potentially illuminable voxels.

14. The method of claim **13** further comprising mapping said position specifiers into lighting plane positions.

15. The method of claim **14** further comprising associating a reference, identifying a voxel specified by said voxel position specifier, with a lighting plane position determined by said mapping, when a reference has not already been associated with said lighting plane position.

16. The method of claim **15** wherein associating comprises loading a priority table operable to associate lighting plane positions with a plurality of said references.

17. The method of claim **1** further comprising producing a priority table operable to associate lighting plane positions with priority references to voxels that map to said lighting plane positions, with priority to references to voxels that are in more foreground planes relative to a lighting plane, over references to voxels that are in more background planes relative to the lighting plane.

18. The method of claim **17** further comprising modifying lighting values associated with voxels identified by said priority references in said priority table.

19. The method of claim **18** wherein modifying said lighting values comprises applying said specular and diffuse illumination values to change an intensity value associated with a voxel identified by a priority reference in said priority table.

20. The method of claim **19** wherein modifying said lighting values comprises locating said specular and diffuse illumination values in response to a normal vector associated with said voxel identified by said priority reference.

21. The method of claim **20** wherein locating comprises using said normal vector as an index to a table associating normal vectors with said specular and diffuse illumination values.

22. The method of claim **17** further comprising receiving a reference identifying a voxel upon which a lighting effect is to be produced.

23. The method of claim **22** further comprising mapping a voxel identified by a received reference into a lighting plane position.

24. The method of claim **23** further comprising addressing said priority table to find a priority reference in said priority table, associated with said lighting plane position.

25. The method of claim **24** further comprising modifying a lighting intensity value associated with said voxel identified by said priority reference, using said specular and diffuse lighting values, when said priority reference associated with said lighting plane matches said received reference.

26. The method of claim **25** wherein modifying said lighting values comprises locating said specular and diffuse illumination values in response to a normal vector associated with said voxel identified by said received reference.

27. The method of claim **26** wherein locating comprises using said normal vector as an index to a table associating normal vectors with said specular and diffuse illumination values.

28. A method of producing a lighting effect on voxel data, the method comprising:

identifying renderable voxels of said voxel block which are more foreground than other renderable voxels in said voxel block, relative to a lighting plane; and

modifying pre-stored lighting values associated with identified voxels, according to pre-computed specular and diffuse illumination values associated with normal vectors of said voxels.

29. A method of associating lighting plane positions on a lighting plane, with voxels in a voxel block, for use in determining which voxels are to receive lighting effects, the method comprising:

scanning voxel positions in said voxel block, according to a scan order in which column positions, in respective planes through said voxel block which are more foreground relative to said lighting plane are scanned before those which are more background relative to said lighting plane, to produce position specifiers identifying illuminable voxel positions in said voxel block;

determining respective lighting plane positions for said position specifiers according to a lighting plane position mapping;

associating respective lighting plane positions with references to voxels identified by said position specifiers such that said lighting plane positions which are already associated with references do not receive new references when a position specifier maps to a lighting plane position to which a reference has already been mapped; and

maintaining a record of said lighting plane positions and their associated references, said references identifying voxels to which lighting effects may be applied.

30. A computer readable medium for providing computer readable instructions for directing a processor circuit to:

compute specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel; and

associate said specular and diffuse illumination values with respective said vectors to facilitate retrieval of said specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced.

31. A signal comprising computer readable code segments comprising:

a first code segment for directing a processor circuit to compute specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel; and

a second code segment for directing a processor circuit to associate said specular and diffuse illumination values with respective said vectors to facilitate retrieval of said specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced.

32. An apparatus for producing a lighting effect on voxel data, the apparatus comprising:

means for computing specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel; and

means for associating said specular and diffuse illumination values with respective said vectors to facilitate retrieval of said specular and diffuse illumination values associated with a normal vector of a voxel upon which a lighting effect is to be produced.

33. An apparatus for producing a lighting effect on voxel data, the apparatus comprising:

a processor circuit operable to compute specular and diffuse illumination values for respective vectors of a set of unit normal vectors representing a plurality of possible different directions away from any voxel in a voxel block; and

memory accessible by said processor circuit to enable said processor circuit to associate said specular and diffuse illumination values with respective said unit normal vectors to facilitate retrieval of said specular and diffuse illumination values associated with a unit normal vector of a voxel upon which the lighting effect is to be produced.

34. The apparatus of claim **33** wherein said processor circuit is programmed to compute a direction of a unit half-way vector H at an angle half-way between a unit illumination vector L representing a direction to a lighting plane and a viewing vector V representing a direction to a view plane.

35. The apparatus of claim **34** wherein said processor circuit is programmed to compute a table associating a plurality of quantized unit normal vectors with corresponding specular illumination values calculated according to the relation:

$$S = (N \bullet H)^k$$

Where S is a calculated specular illumination value

N is a quantized unit normal vector from said set of unit normal vectors

H is said unit half-way vector between said viewing vector V and said unit illumination vector L, and

k is an exponent representing a property of a surface

36. The apparatus of claim **35** wherein said processor circuit is programmed to produce a table associating a plurality of quantized unit normal vectors with corresponding diffuse illumination values calculated according to the relation:

$$D = N \bullet L$$

Where D is a calculated specular illumination value

N is a quantized unit normal vector from said set of unit normal vectors; and

L is said unit illumination vector

37. The apparatus of claim **34** wherein said processor circuit is operable to produce said diffuse and specular illumination values by negating L when the dot product of V and L vectors is negative.

38. The apparatus of claim **36** wherein said processor circuit is operable to apportion said tables into portions associated with respective areas of a sphere within which said unit normal vectors may lie.

39. The apparatus of claim **33** further comprising an identifier in communication with said processor circuit and operable to identify a voxel upon which the lighting effect is to be produced.

40. The apparatus of claim **39** wherein said identifier comprises said processor circuit.

41. The apparatus of claim **39** further comprising a modifier operable to communicate with said memory to obtain said specular and diffuse illumination values and operable to access and modify prestored lighting values associated with an identified voxel according to at least one of said specular and diffuse illumination values determined using a unit normal vector of said identified voxel.

42. The apparatus of claim **41** wherein said modifier comprises said processor circuit.

43. The apparatus of claim **41** wherein said identifier further comprises a scanner operable to scan said voxel block.

44. The apparatus of claim **39** wherein said identifier is operable to associate a reference identifying a voxel specified by a voxel position specifier, with a lighting plane position determined by a mapper, when a reference has not already been associated with said lighting plane position.

45. The apparatus of claim **44** further comprising a mapper for mapping said position specifiers produced by said identifier into lighting plane positions.

46. The apparatus of claim **45** further comprising a scanner operable to scan said voxel block according to a scan order in which column positions in respective planes through said voxel block that are progressively further from a lighting plane are addressed in order of those that map to foreground positions in the lighting plane before those that map to more background positions in the lighting plane, to produce said voxel position specifiers identifying potentially illuminable voxels.

47. The apparatus of claim **46** wherein said scanner comprises said processor circuit.

48. The apparatus of claim **46** further comprising memory accessible by said scanner for storing a priority table operable to associate lighting plane positions with respective said references, with priority given to references to voxels that are in more foreground planes relative to a lighting plane, over references to voxels that are in more background planes relative to said lighting plane.

49. The apparatus of claim **48** further comprising a modifier in communication with said memory storing said priority table and operable to modify lighting values associated with voxels identified by said references in said priority table.

50. The apparatus of claim **49** wherein said modifier is operable to apply said specular and diffuse illumination values to change an intensity value associated with a voxel identified by a reference in said priority table.

51. The apparatus of claim **51** wherein said modifier is operable to locate said specular and diffuse illumination values in said memory in response to a unit normal vector associated with said voxel identified by said reference.

52. The apparatus of claim **41** further comprising a receiver in communication with said modifier, operable to receive a reference identifying a voxel upon which the lighting effect is to be produced.

53. The apparatus of claim **52** further comprising a mapper operable to map a voxel identified by a received reference into a lighting plane position.

54. The apparatus of claim **53** further comprising a memory accessible by said modifier, for storing a priority table, said modifier being operable to address said priority table to find a reference associated with said lighting plane position, in said priority table.

55. The apparatus of claim **54** wherein said modifier is operable to modify a lighting intensity value associated with said voxel identified by said reference, using said specular and diffuse lighting values, when said reference associated with said lighting plane matches said received reference.

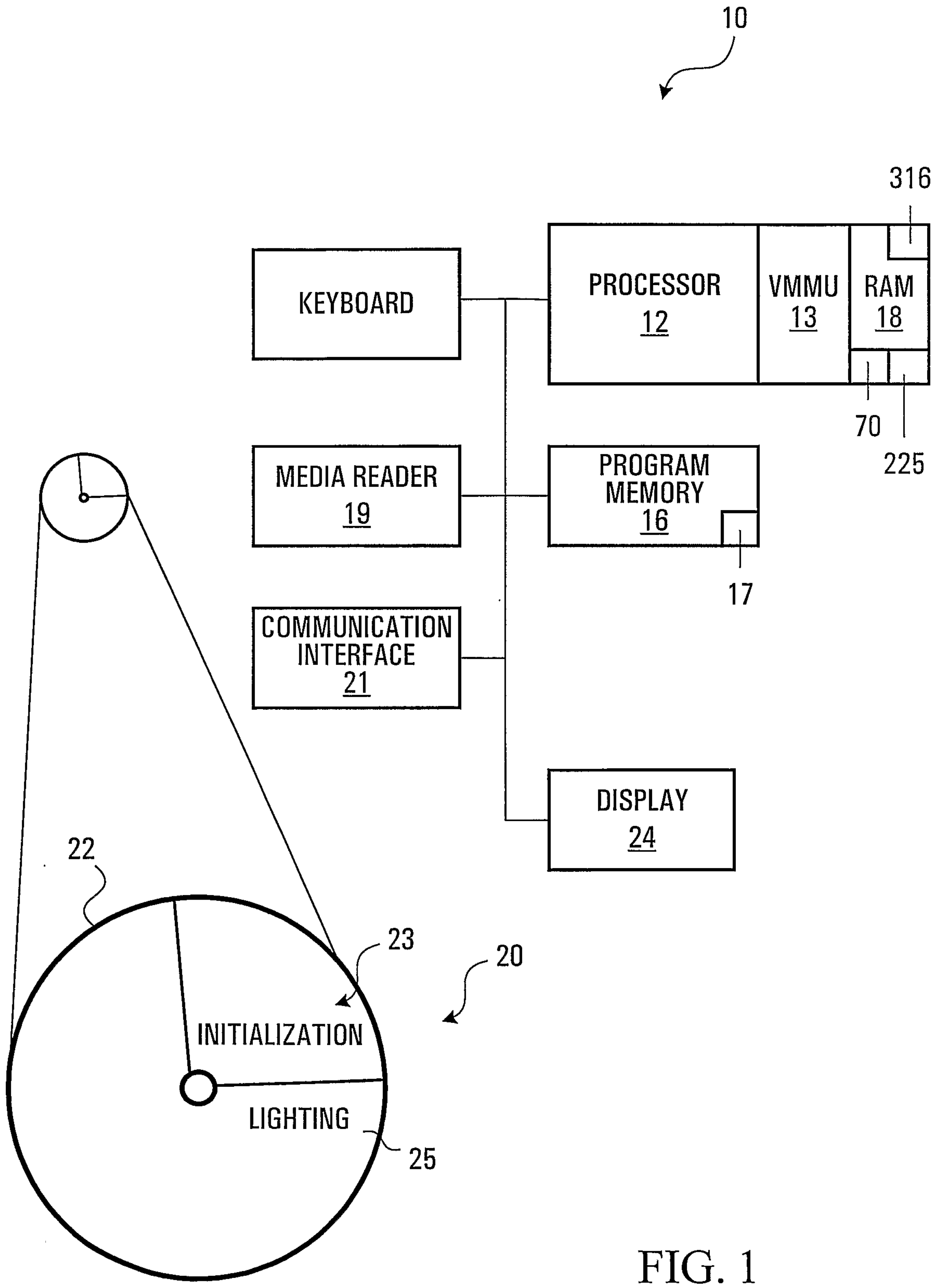

10
KEYBOARD
PROCESSOR 12
VMMU 13
RAM 18
316
70
225
MEDIA READER 19
PROGRAM MEMORY 16
17
COMMUNICATION INTERFACE 21
DISPLAY 24
22
23
20
INITIALIZATION
LIGHTING
25

FIG. 1

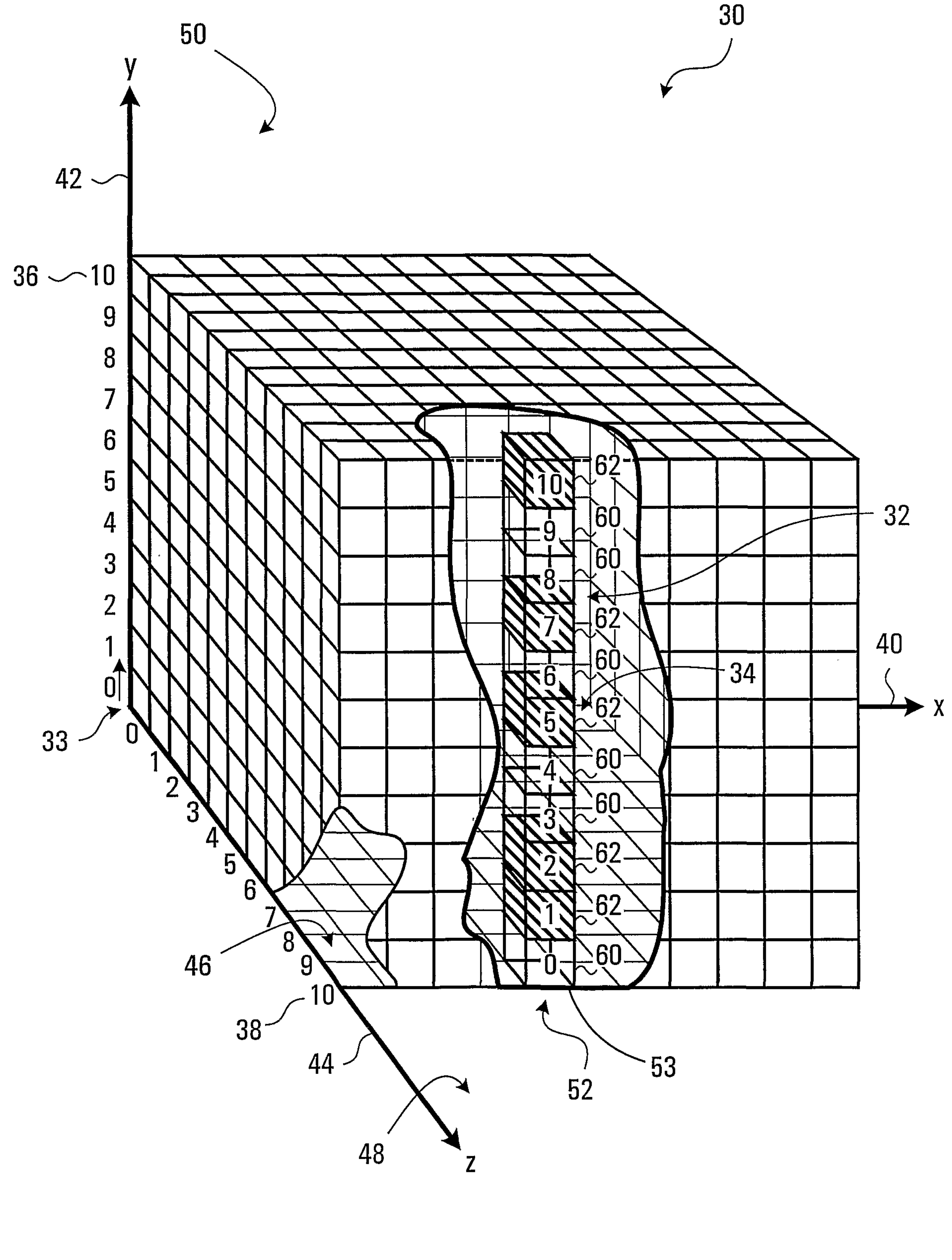

30
50
y
42
36
10
9
8
7
6
5
4
3
2
1
0
33
0
1
2
3
4
5
6
7
8
9
10
38
44
46
48
z
62
10
60
9
60
8
32
62
7
60
6
34
62
5
60
4
60
3
62
2
62
1
60
0
40
x
52
53

FIG. 2

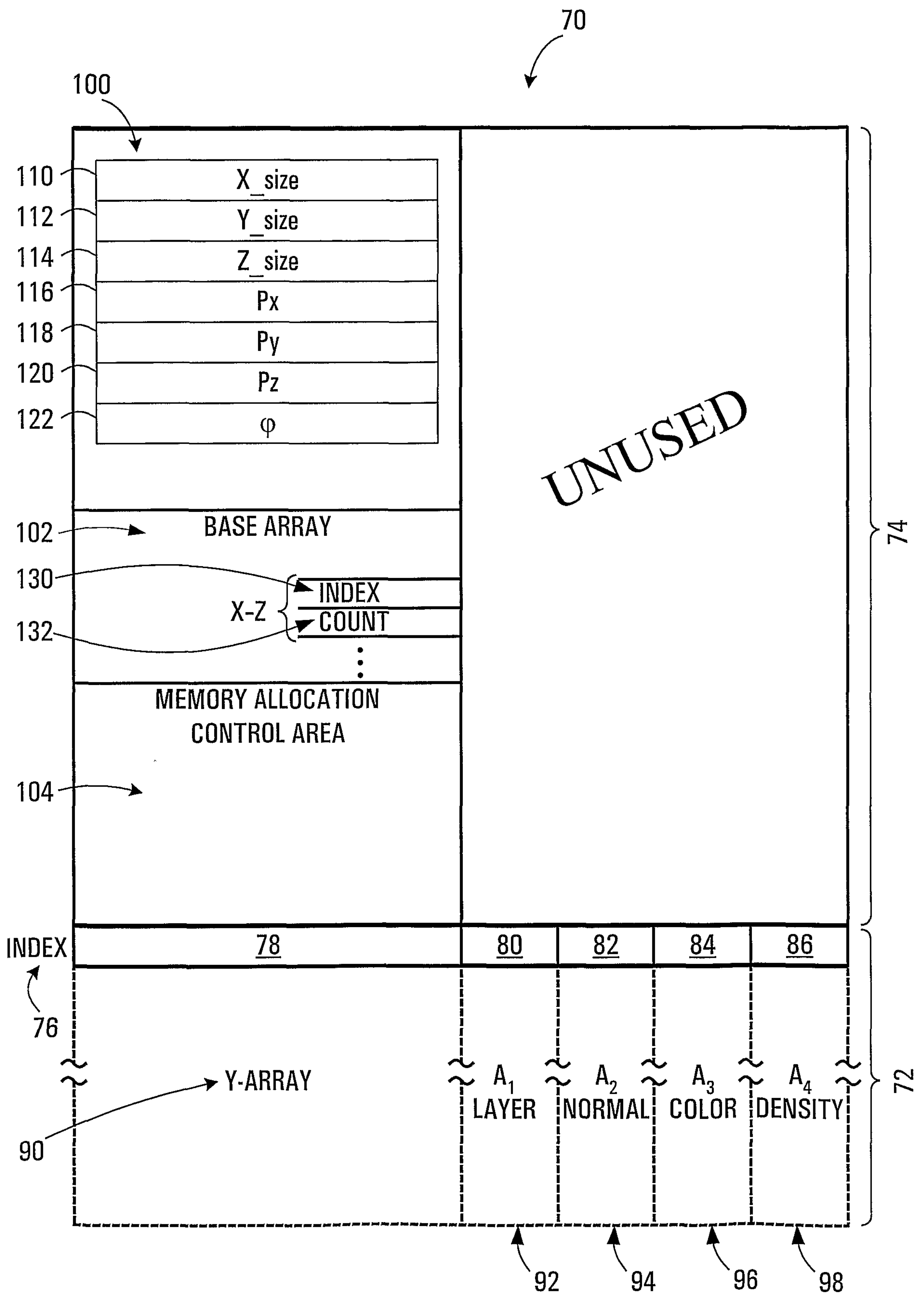

70
100
110
112
114
116
118
120
122
X_size
Y_size
Z_size
Px
Py
Pz
φ
UNUSED
102
BASE ARRAY
130
132
X-Z
INDEX
COUNT
MEMORY ALLOCATION CONTROL AREA
104
74
INDEX
78
80
82
84
86
76
Y-ARRAY
90
A_1 LAYER
A_2 NORMAL
A_3 COLOR
A_4 DENSITY
72
92
94
96
98

FIG. 3

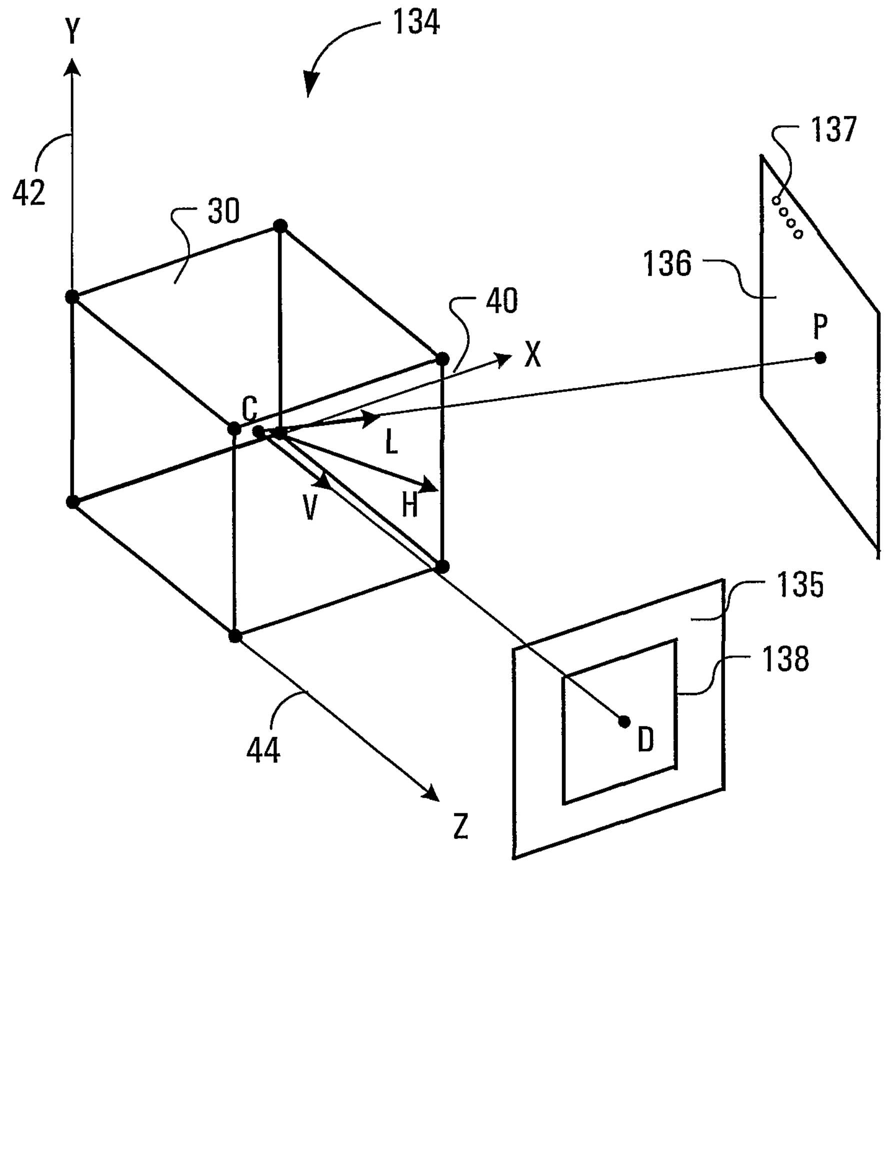


FIG. 4

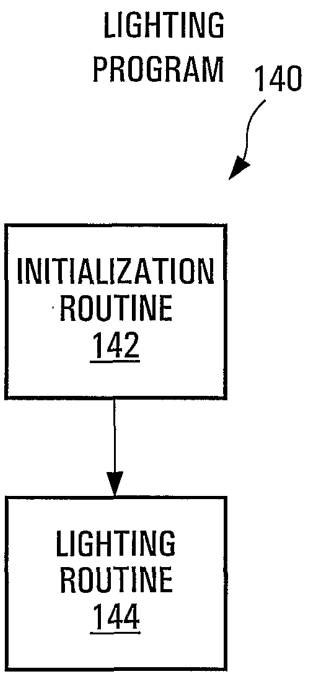

LIGHTING PROGRAM
140
INITIALIZATION ROUTINE
142
LIGHTING ROUTINE
144

FIG. 5

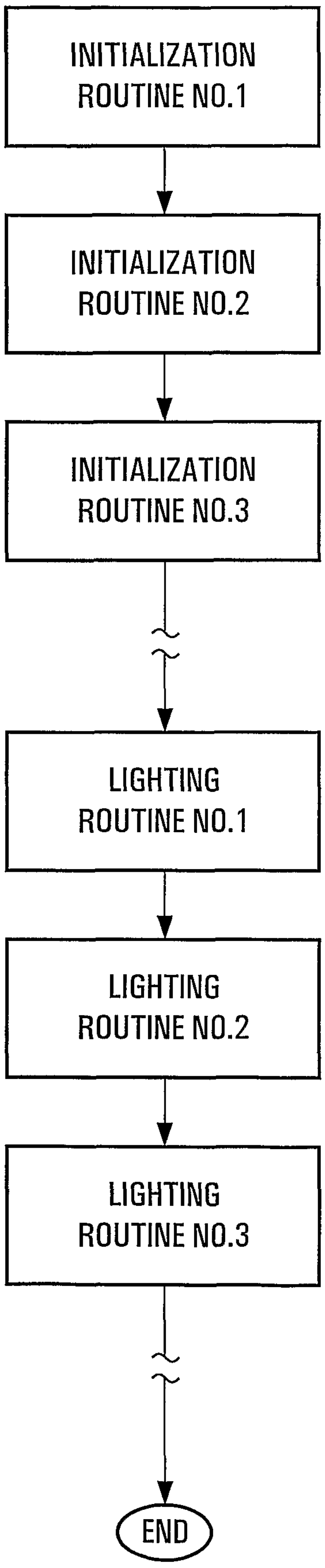

INITIALIZATION ROUTINE NO.1
INITIALIZATION ROUTINE NO.2
INITIALIZATION ROUTINE NO.3
LIGHTING ROUTINE NO.1
LIGHTING ROUTINE NO.2
LIGHTING ROUTINE NO.3
END

FIG. 6

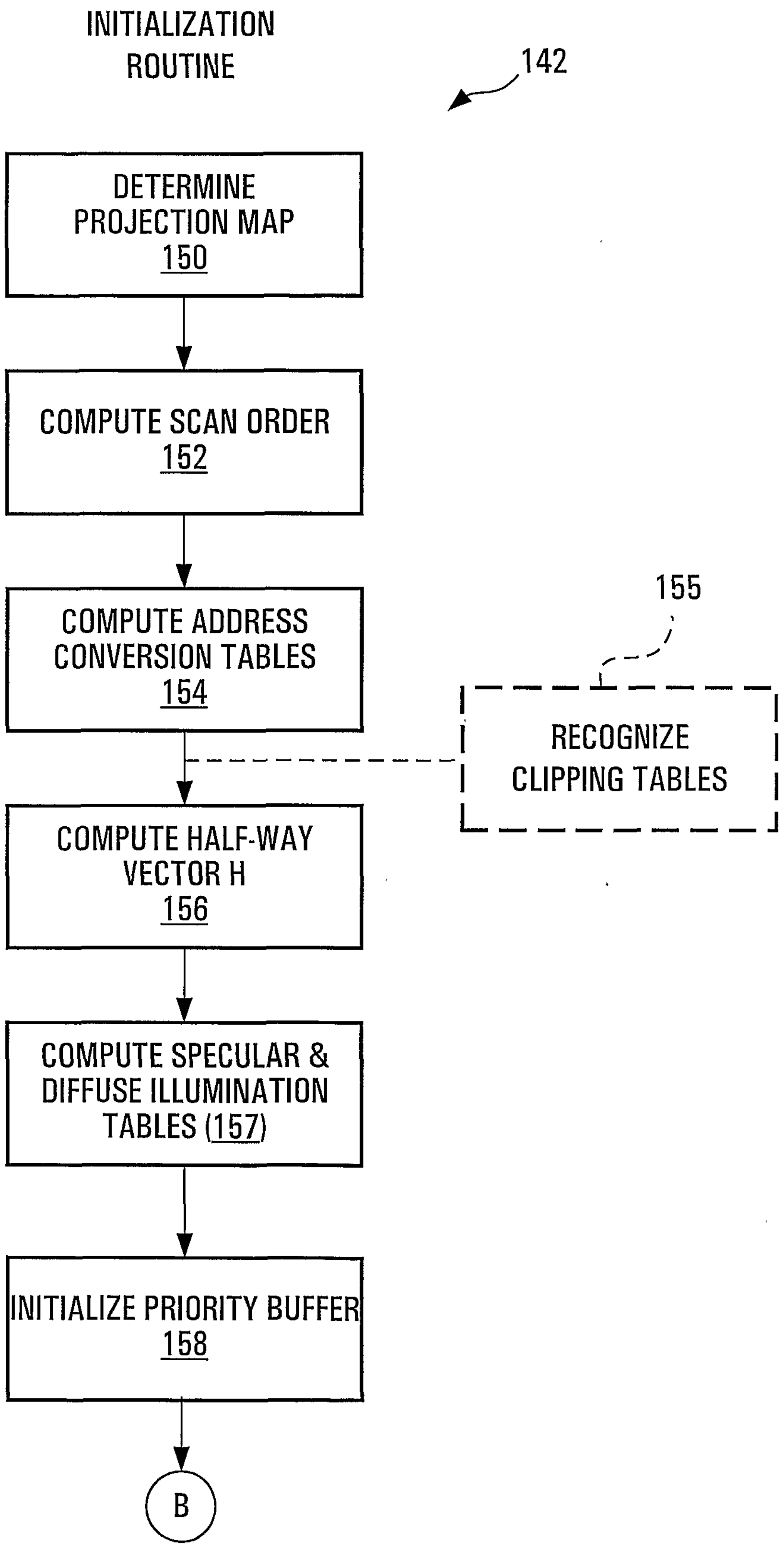

INITIALIZATION ROUTINE
142
DETERMINE PROJECTION MAP 150
COMPUTE SCAN ORDER 152
COMPUTE ADDRESS CONVERSION TABLES 154
155
RECOGNIZE CLIPPING TABLES
COMPUTE HALF-WAY VECTOR H 156
COMPUTE SPECULAR & DIFFUSE ILLUMINATION TABLES (157)
INITIALIZE PRIORITY BUFFER 158
B

FIG. 7A

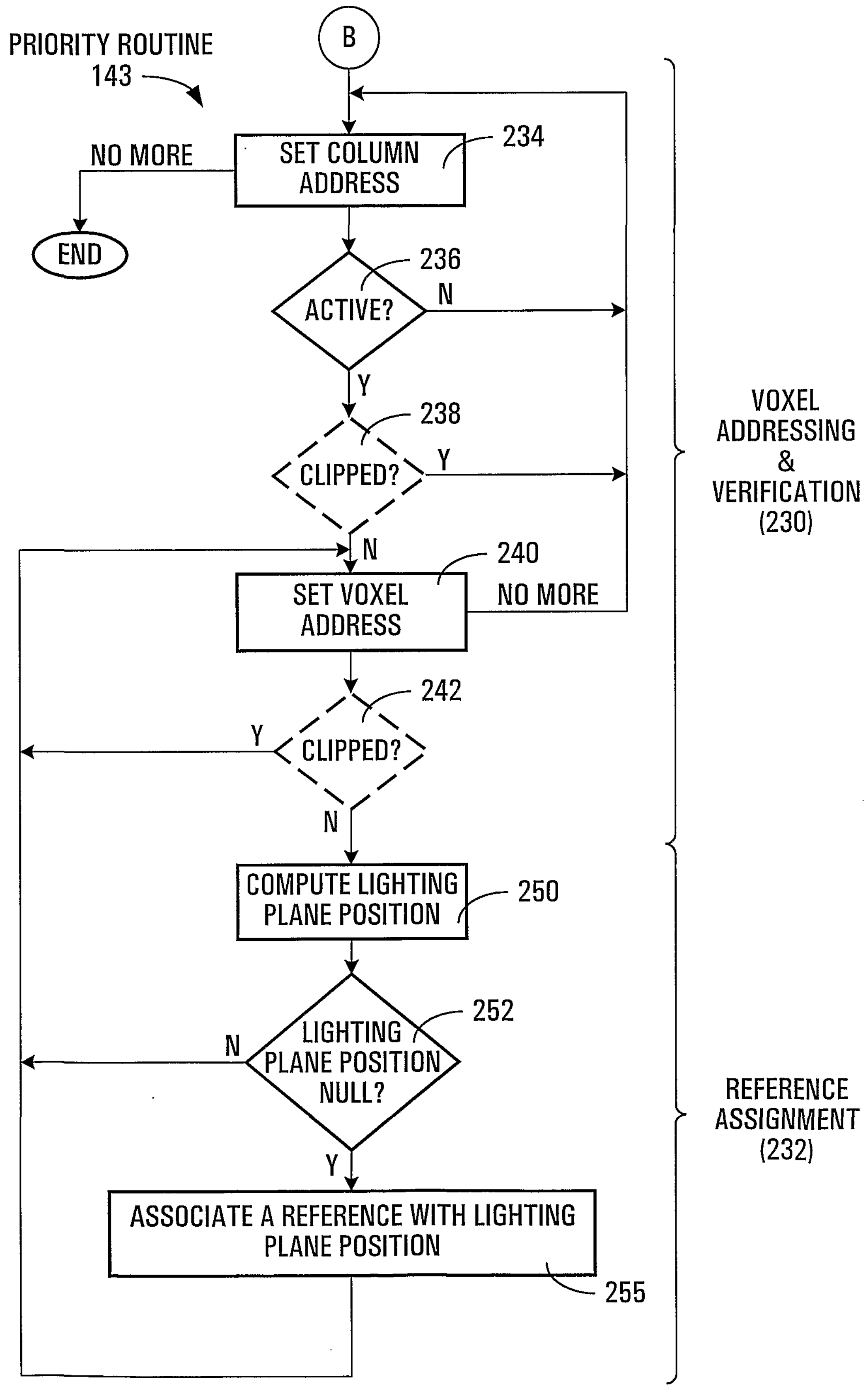

PRIORITY ROUTINE
143
B
SET COLUMN ADDRESS
234
NO MORE
END
ACTIVE?
236
N
Y
CLIPPED?
238
Y
N
SET VOXEL ADDRESS
240
NO MORE
CLIPPED?
242
Y
N
COMPUTE LIGHTING PLANE POSITION
250
LIGHTING PLANE POSITION NULL?
252
N
Y
ASSOCIATE A REFERENCE WITH LIGHTING PLANE POSITION
255
VOXEL ADDRESSING & VERIFICATION (230)
REFERENCE ASSIGNMENT (232)

FIG. 7B

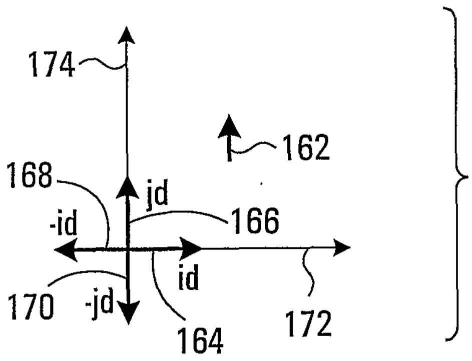


FIG. 8

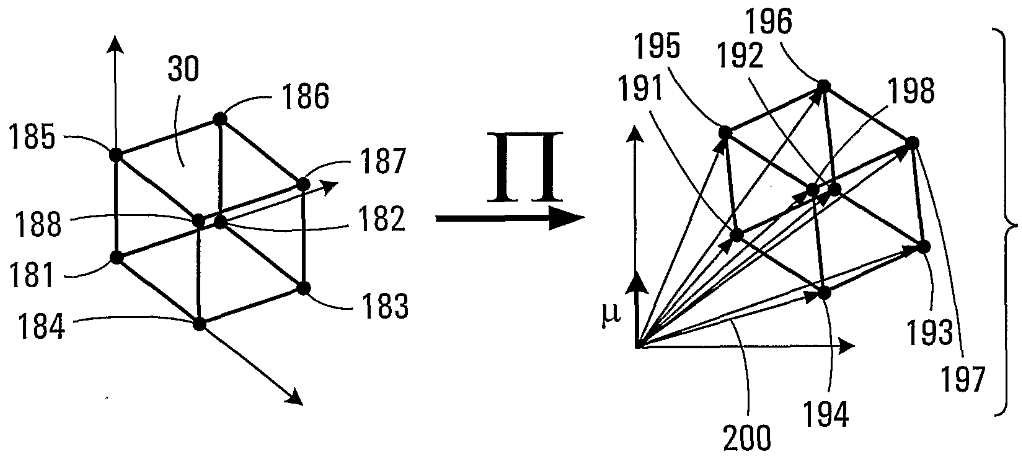


FIG. 9

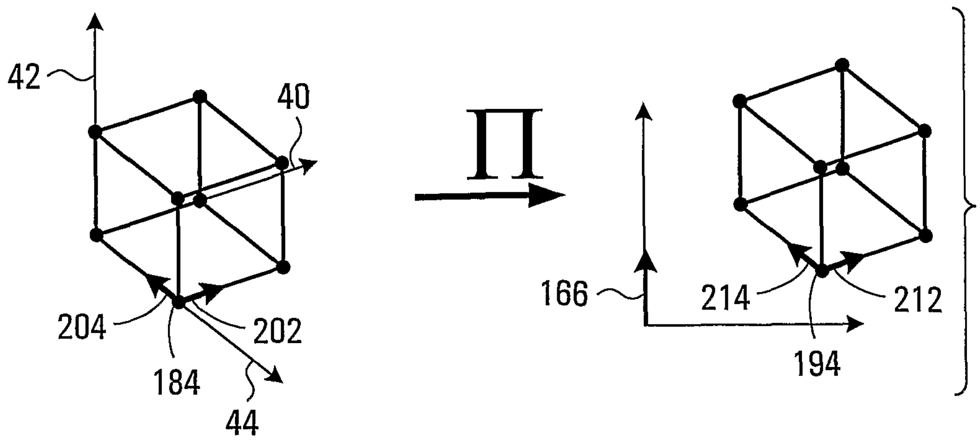


FIG. 10

PLANE (Z)	COLUMN (X)	Y DIRECTION(Y)
0- $V_P(\Pi(0,0,1))$	0- $V_C(\Pi(1,0,0))$	0- $V_Y(\Pi(0,1,0))$
1- $V_P(\Pi(0,0,1))$	1- $V_C(\Pi(1,0,0))$	1- $V_Y(\Pi(0,1,0))$
2- $V_P(\Pi(0,0,1))$	2- $V_C(\Pi(1,0,0))$	2- $V_Y(\Pi(0,1,0))$
⋮	⋮	⋮
Z_Size- $V_P(\Pi(0,0,1))$	X_size- $V_c(\Pi(1,0,0))$	Y_size- $V_Y(\Pi(0,1,0))$

FIG. 11

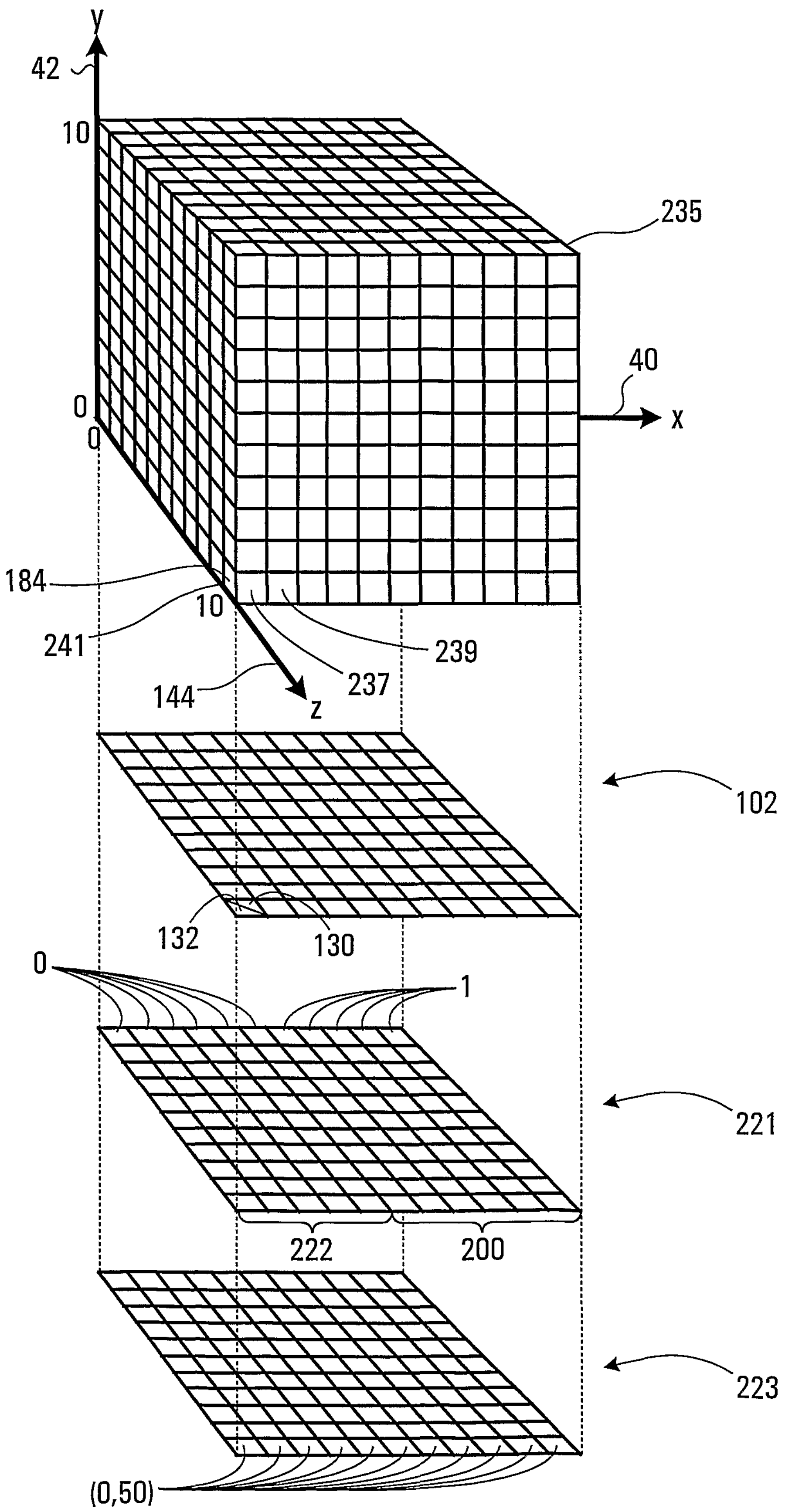

y
42
10
0
0
235
40
x
184
10
241
239
237
144
z
102
132
130
0
1
221
222
200
223
(0,50)

FIG. 12

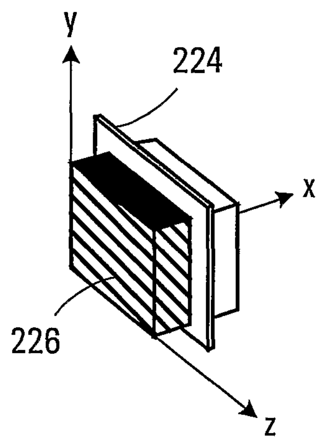


FIG. 13

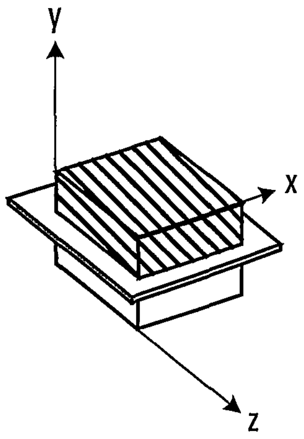


FIG. 14

SPECULAR ILLUMINATION TABLE

161

N	$S = (N \bullet H)^k$
0	nnn
1	nnn
2	nnn
3	nnn
.	.
.	.
.	.
4095	nnn
4096	nnn

167 165

DIFFUSE ILLUMINATION TABLE

163

N	$D = N \bullet L$
0	nnn
1	nnn
2	nnn
3	nnn
.	.
.	.
.	.
4095	nnn
4096	nnn

167 169

FIG. 15

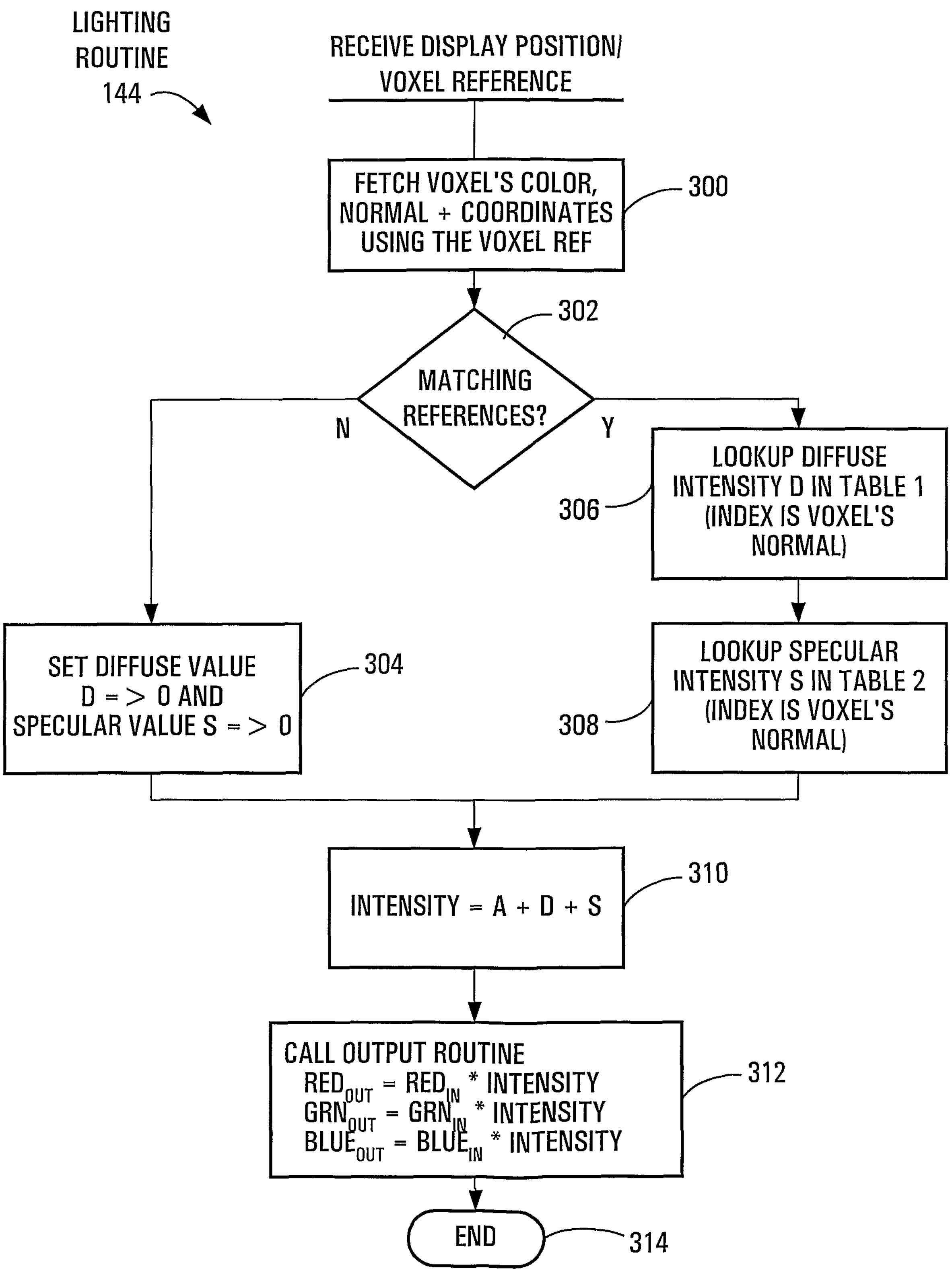

LIGHTING ROUTINE 144
RECEIVE DISPLAY POSITION/ VOXEL REFERENCE
FETCH VOXEL'S COLOR, NORMAL + COORDINATES USING THE VOXEL REF
300
MATCHING REFERENCES?
302
N
Y
LOOKUP DIFFUSE INTENSITY D IN TABLE 1 (INDEX IS VOXEL'S NORMAL)
306
SET DIFFUSE VALUE D => 0 AND SPECULAR VALUE S => 0
304
LOOKUP SPECULAR INTENSITY S IN TABLE 2 (INDEX IS VOXEL'S NORMAL)
308
INTENSITY = A + D + S
310
CALL OUTPUT ROUTINE
$RED_{OUT} = RED_{IN} * INTENSITY$
$GRN_{OUT} = GRN_{IN} * INTENSITY$
$BLUE_{OUT} = BLUE_{IN} * INTENSITY$
312
END
314

FIG. 16

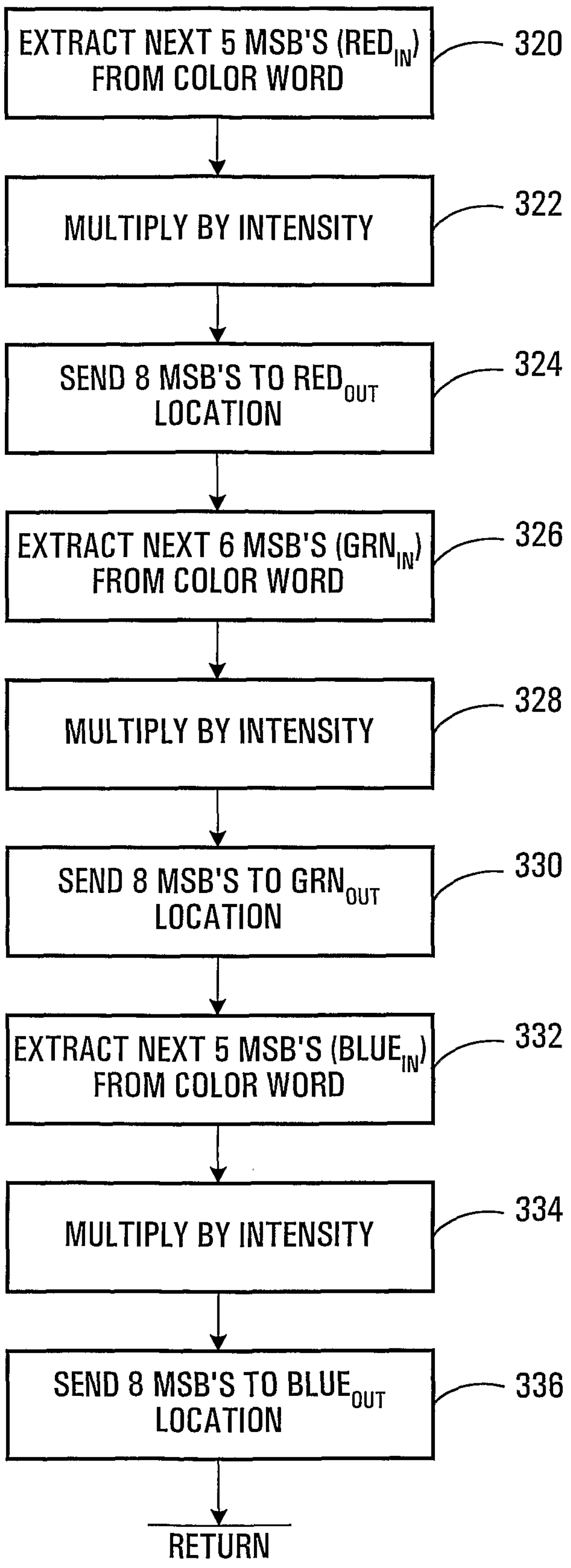

EXTRACT NEXT 5 MSB'S (RED_{IN}) FROM COLOR WORD
320
MULTIPLY BY INTENSITY
322
SEND 8 MSB'S TO RED_{OUT} LOCATION
324
EXTRACT NEXT 6 MSB'S (GRN_{IN}) FROM COLOR WORD
326
MULTIPLY BY INTENSITY
328
SEND 8 MSB'S TO GRN_{OUT} LOCATION
330
EXTRACT NEXT 5 MSB'S ($BLUE_{IN}$) FROM COLOR WORD
332
MULTIPLY BY INTENSITY
334
SEND 8 MSB'S TO $BLUE_{OUT}$ LOCATION
336
RETURN

FIG. 17

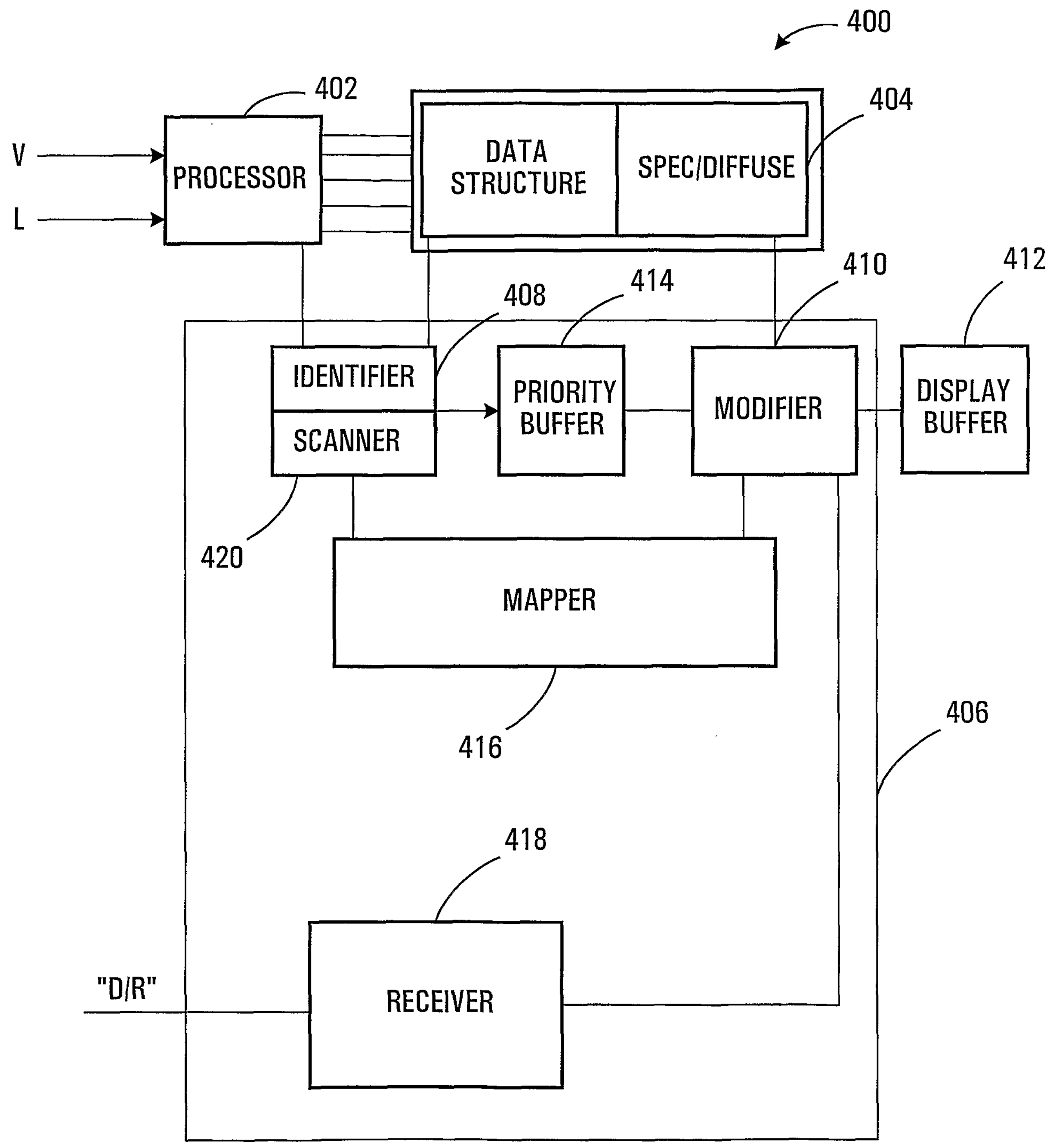

400
402
V
L
PROCESSOR
DATA STRUCTURE
SPEC/DIFFUSE
404
408
414
410
412
IDENTIFIER
SCANNER
PRIORITY BUFFER
MODIFIER
DISPLAY BUFFER
420
MAPPER
416
406
418
"D/R"
RECEIVER

FIG. 18