PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06T 3/00 — A1

(11) International Publication Number: **WO 97/30420**

(43) International Publication Date: 21 August 1997 (21.08.97)

(21) International Application Number: PCT/US97/02290

(22) International Filing Date: 13 February 1997 (13.02.97)

(30) Priority Data:
08/601,596 14 February 1996 (14.02.96) US

(71) Applicant: DIGITAL MEDIA INTERACTIVE [US/US]; Suite 215, 1730 South Amphlett Boulevard, San Mateo, CA 94402 (US).

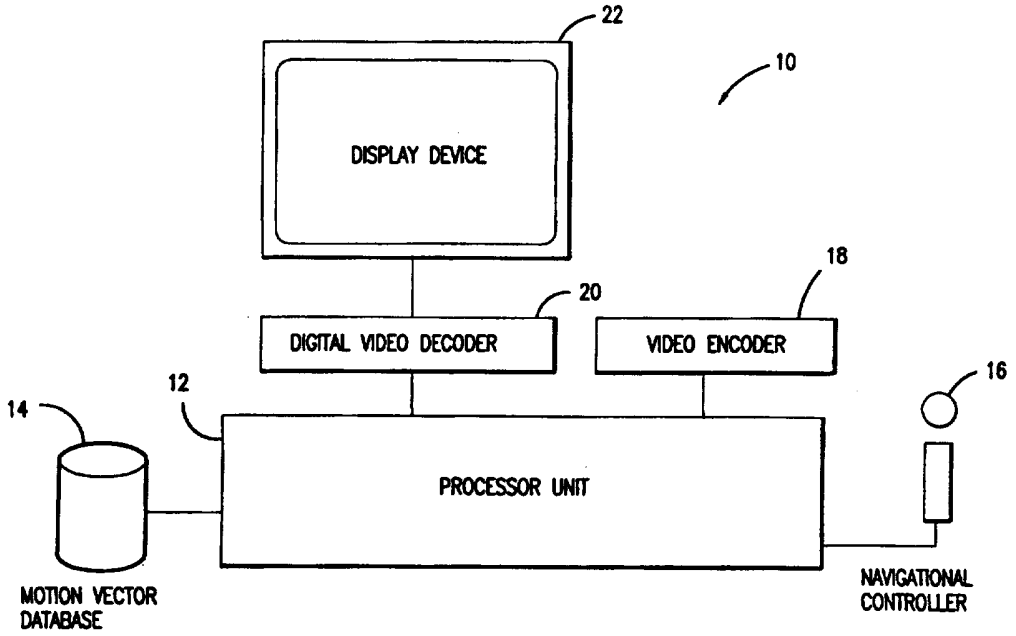
(72) Inventor: SEGAL, Gerald, M.; 1119 Royal Lane, San Carlos, CA 94070 (US).

(74) Agents: MORRIS, Francis, E. et al.; Pennie & Edmonds L.L.P., 1155 Avenue of the Americas, New York, NY 10036 (US).

(81) Designated States: AL, AM, AU, AZ, BA, BB, BG, BR, BY, CA, CN, CU, CZ, EE, GE, HU, IL, IS, JP, KG, KP, KR, KZ, LC, LK, LR, LT, LV, MD, MG, MK, MN, MX, NO, NZ, PL, RO, RU, SG, SI, SK, TJ, TM, TR, TT, UA, UZ, VN, YU, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD AND SYSTEM FOR THE CREATION OF AND NAVIGATION THROUGH A MULTIDIMENSIONAL SPACE USING ENCODED DIGITAL VIDEO



(57) Abstract

The present invention provides a method and system for the creation of, and navigation through, a multidimensional virtual space (50) using digital video encoding (18) and decoding (20) technique. Specifically, the use of a novel database of randomly accessible, pre-rendered short video sequences associated with a set of predefined "motion vectors" (14) allows a user or operator to navigate smoothly through a highly realistic virtual space.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

METHOD AND SYSTEM FOR THE CREATION OF AND NAVIGATION THROUGH A MULTIDIMENSIONAL SPACE USING ENCODED DIGITAL VIDEO

FIELD OF THE INVENTION

The present invention relates generally to computer graphics, and more particularly to a method and system for the creation and navigation of a multidimensional space using encoded digital video.

BACKGROUND

Most currently available virtual reality or interactive computer systems (such as video games) create images by generating, or "rendering" a plurality of polygons in real time. These rendered polygons are displayed on a screen, and together form a "scene".

Such systems typically allow the user or operator to "move" through the scene and to view various scenes by manipulating a pointing or positioning device such as a joystick or track ball. Input from the pointing or positioning device causes the computer system to calculate the appropriate change in position and, using a three-dimensional mathematical model (often called a "parametric" model) of the objects in the virtual space, render a new scene in real time. An illusion of motion is created by sequentially displaying a series of images that change appropriately in accordance with the inputs from the pointing or positioning device. An example of a commercially available virtual reality package incorporating parametric

models is the Virtual Reality Development System manufactured by VREAM, Inc. of Chicago, Illinois.

Typical virtual reality or interactive computer systems are limited in the amount of detail or realism they can display because of the large amount of computing power required to render a realistic scene in real time. The vast majority of such systems can only display VGA or slightly better than VGA graphics. Specular highlights, texture mapping, shadows, and other rendering techniques associated with the better rendering and 3D modeling packages (such as 3D Studio®, manufactured by AutoDesk Corporation of Novato, California) are not currently possible in real time on desktop systems, even those incorporating graphics accelerator chips. Moreover, even very powerful and expensive systems, such as supercomputers or very high-end graphics workstations, face limitations in delivering highly realistic scenes in real time because of the computational requirements of rendering such complex scenes.

Although more powerful rendering engines, and more efficient rendering and modeling software are introduced with some frequency, the computational requirements for real time, highly realistic virtual reality or interactive computer applications are still too great for most commonly available computers.

In an attempt to overcome the problems with real time rendering, photographic technologies have been proposed and created that use digitally altered photographs. These technologies, such as QuickTime VR, manufactured by Apple

Computer, Inc. of Cupertino, California, allow the viewer to experience a sense of panorama in viewing a scene. However, most such systems place severe limitations on the size, aspect ratio, and color palette that can be supported for real-time playback.

Recently, great advances have been made in the implementation and standardization of certain digital video compression techniques. The Moving Picture Experts Group (MPEG) was chartered by the International Standards Organization (ISO) to standardize a coded representation of video (and associated audio) suitable for digital storage and transmission media. Digital storage media include magnetic computer disks, optical compact disk read-only-memory (CD-ROM), digital audio tape (DAT), etc. Transmission media include telecommunications networks, home coaxial cable TV (CATV), over-the-air digital video, and other media. The goal of MPEG has been to develop a generic coding standard that can be used in many digital video implementations. MPEG has so far produced two standards, known colloquially as MPEG-1 and MPEG-2.

MPEG-1 (officially known as ISO/IEC 11172) is an international standard for coded representation of digital video and associated audio at bit-rates up to about 1.5 Mbits/s. MPEG-1 can typically provide video compression ratios of between 140:1 and 200:1; it is currently used in relatively limited bandwidth devices, such as CD-ROM players. The ISO/IEC 11172 (MPEG-1) standard is incorporated herein by reference.

MPEG-2 (officially known as ISO/IEC 13818) is a standard for coded representation of digital video and associated audio at bit-rates above 2 Mbits/s. MPEG-2 can typically provide video compression ratios of between 40:1 and 60:1; it is intended for use in relatively high bandwidth devices and broadcast television. The ISO/IEC 13138 (MPEG-2) standard is also incorporated herein by reference.

The MPEG compression techniques are based in part on the fact that in most motion picture or video scenes, the background remains relatively stable while much of the action takes place in the foreground; hence, consecutive frames in a video sequence often contain some identical or very similar image information.

MPEG compression generally begins by the creation of a reference frame or picture called an "I" or "intra" frame. Intra frames provide entry points into an MPEG video sequence file for random access, but can only be moderately compressed. I frames are typically placed every 10 to 15 frames in a video sequence. MPEG compression takes advantage of the redundancy often found in sequential frames of video by capturing, compressing, and storing the differences between a set of sequential frames. The other two types of frames in an MPEG sequence are predicted (P) frames; and bi-directional interpolated (B) frames. Predicted frames are encoded with reference to a past frame (I or previous P frame), and, in general, are used as a reference for future predicted frames. Predicted frames receive a fairly high amount of compression. Bi-directional interpolated frames

provide the highest amount of compression, but require both a past and a future reference in order to be encoded. B frames are never used as references.

The MPEG video standards specify the syntax and semantics of the compressed bit-stream produced by an MPEG video encoder. The standards also specify how this bit-stream is to be parsed and decoded to produce a decompressed video signal. The overall syntax of an MPEG bit-stream is constructed in a hierarchy of several headers, each of which performs a different logical function. For the purposes of this invention, the most important MPEG video bit-stream syntax header is the "Group Of Pictures" (GOP) header. The GOP header provides support for random access, fast search, and editing. A sequence of video frames, or "pictures" is divided into a series of GOPs, where each GOP contains an I frame followed by an arrangement of P frames and B frames. **Figure 1** shows the basic structure of a GOP. Random access and fast search are enabled by the availability of the I frames, which can be decoded independently and serve as starting points for further decoding. The MPEG video standards allow GOPs to be of arbitrary structure and length, and the GOP header is a basic unit for editing an MPEG video bit stream.

Prior systems such as the video game entitled "7th Guest" by the Spectrum Holobyte and Philips Corporation, have used the random access capabilities of MPEG to provide "branching". That is, at a predefined point in the game, an operator may choose from two or more options as to where to

go, or what to do next. However, such prior systems do not allow continuous interactive input to be made by the operator, and thus do not provide a highly realistic virtual reality or interactive computer environment. Accordingly, there remains a need in the art for a highly realistic, relatively low cost virtual reality or interactive computer system that allows for the generation of and real time navigation through a virtual space.

SUMMARY OF THE INVENTION

The present invention meets these needs by providing a method and system for the creation of, and navigation through, a multidimensional virtual space using digital video encoding and decoding techniques. Specifically, the use of a novel database of randomly accessible, pre-rendered short video sequences associated with a set of predefined "motion vectors" allows a user or operator to navigate smoothly through a highly realistic virtual space.

In a preferred embodiment, a three dimensional model of the virtual space is generated (using a three dimensional rendering package, such as 3D Studio®) and imported into the system. A coordinate system, including a plurality of fixed points, is defined and imposed on the three dimensional model. The number of degrees of freedom an operator may have, and the areas constrained from navigation may also be defined.

Based on the coordinate system chosen and the geometry of the virtual space, the system creates a database of motion vectors. These motion vectors represent the virtual path followed by an operator when moving from point to point, or when rotating about a fixed point in the virtual space. A separate motion vector is generated for each allowed movement (that is, translation or rotation) within the virtual space.

Each motion vector is associated with a prerendered video sequence which shows the translation or rotation represented by that motion vector. After the translation or rotation represented by each motion vector has been stored in the database, the video sequences associated with the motion vectors are rendered, encoded, and stored. In a preferred embodiment, the system automatically generates instructions for a renderer as to the position and movement of a "virtual camera" through the virtual space. These instructions, called "camera path data", indicate to the renderer which viewpoints and video sequences (associated with the previously defined motion vectors) should be rendered and encoded. Each video sequence associated with a motion vector is preferably a relatively short (for example, eight to fifteen frame) sequence which corresponds to an encoded MPEG Group of Pictures (GOP).

To allow navigation through the virtual space, a state machine is created which keeps track of the position, direction, and motion of an operator navigating through the virtual space. Based upon the position and direction of the

operator in the virtual space, the state machine dictates which motion vector (more specifically, which video sequence associated with a particular motion vector) should be displayed. The state machine reflects the number of degrees of freedom an operator may have, and the navigational characteristics of the controller device (such as a joystick or track ball) used to navigate through the virtual space.

As an operator moves through the virtual space, the state machine is used to play the appropriate prerendered video sequences through a "virtual projector". The virtual projector cues and plays the prerendered video sequences according to the instructions of the state machine, overlapping in time the video sequence boundaries in such a manner that the displayed video sequences approximate smooth linear motion. In a preferred embodiment, the last frame of a video sequence associated with a particular motion vector is identical to the first frame in a video sequence associated with an adjoining motion vector.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a representational diagram showing the general organization of frames in an MPEG video sequence.

Figure 2 is a block diagram showing a system for generating a database of motion vectors and their associated video sequences according to a preferred embodiment.

Figure 3 shows a sample reference grid within a virtual space according to a preferred embodiment.

Figure 4 is a representation of a motion vector in a reference grid according to a preferred embodiment.

Figure 5 is a sample listing of camera path data according to a preferred embodiment.

Figures 6A - 6D are illustrative diagrams showing possible orientations and motions of an operator at a particular point in a virtual space.

Figures 7A - 7D are examples of state tables as used by a state machine to determine the motion and orientation of an operator navigating through a virtual space.

Figure 8 is a block diagram of a playback system according to a preferred embodiment.

Figure 9 is a diagram conceptually illustrating the use of virtual projectors according to a preferred embodiment.

Figure 10 is a diagram conceptually illustrating a motion vector probability envelope.

Figure 11 is a diagram showing the control and operation of virtual projectors according to a preferred embodiment.

DETAILED DESCRIPTION

The structure and function of the preferred embodiments can best be understood by reference to the drawings. The reader will note that the same reference numerals appear in multiple figures. Where this is the case, the numerals refer to the same or corresponding structure in those figures.

The preferred embodiments of the present invention include a method and system for the creation of and navigation through a multidimensional space using encoded digital video. A concatenated series of prerendered video sequences (which are associated with predefined "motion vectors") allow navigation through the space. The video sequences are preferably encoded according to the MPEG-1 or MPEG-2 standards discussed above, although other randomly accessible, digital video encoding techniques developed either now or in the future could also be used to encode the video sequences.

The following discussion will be divided generally into two parts: (1) the methods and systems used to create a multidimensional virtual space and the framework needed to navigate within that space, including the use of a coordinate system and the creation of a database of motion vectors and their associated video sequences; and (2) the methods and

systems used by an operator to navigate through the virtual space.

Development of the Virtual Space and the Navigational Framework

The first step in creating a navigational system and framework for a virtual space is the creation of the space itself. The virtual space is preferably created using a general purpose computer, such as an IBM-compatible personal computer. The operation and architecture of such computers is well known in the art, and is explained in several references, including The Winn L. Rosch Hardware Bible (second edition), which is incorporated herein by reference. If desired, the virtual space could also be created using more powerful general purpose computers or workstations, or using dedicated computers.

In a preferred embodiment, the general purpose computer used to create the virtual space runs a three dimensional rendering package such as 3D Studio®, manufactured by Autodesk Corporation. Such commercially available rendering packages allow the creation and rendering of highly realistic three dimensional spaces. A virtual space is created by defining the shape and layout of the space, and by defining the shape and location of objects within the virtual space. In a preferred embodiment, the actual rendering (that is, the creation of full images) of the virtual space takes place in a later step, which will be discussed below.

In an alternative embodiment, a real space (such as a building, racetrack, etc.), could be chosen as the basis for the virtual space navigated by an operator. In this alternative embodiment, which will be discussed further below, real video footage (instead of computer rendered images) could be used to show portions of or scenes within the virtual space.

After the virtual space itself has been generated, it is preferably imported into a development system **10** as shown in **Figure 2**. Development system **10** includes a processor unit **12**, a motion vector array or database **14**, a navigational controller **16**, a digital video encoder **18**, a digital video decoder **20**, and a display device **22**. The interaction between the components of development system **10** is controlled by processor unit **12**, which preferably is or includes a general purpose computer, such as an IBM-compatible personal computer. Again, the operation and architecture of such computers is well known in the art.

Since processor **12** may include a general purpose computer, the rendering package used to generate the three dimensional virtual space may be run on development system **10** if desired. Video encoder **18** and video decoder **20** are preferably MPEG-1 or MPEG-2 type devices, the operation of which are known to those skilled in the art. Navigational controller **16** may be any type of input device, such as a joystick, track ball, keyboard, etc. Display device **22** may be any type of video monitor or video display device. The motion vector database **14** is preferably constructed according

to a network node model that allows motion vectors to be located very quickly.

After a three dimensional model of a virtual space is imported into (or generated by) development system 10, a coordinate system is defined and imposed on the three dimensional model. Alternatively, a coordinate system may have been imposed on the three dimensional model by the rendering package used to generate the model.

For the purposes of explanation here, a rectangular, or cartesian coordinate system is used. However, it will be understood that a spherical, cylindrical, polar, or any other type of coordinate system may be used with the present invention. In a preferred embodiment, the coordinate system is divided into a plurality of discrete node points, which may be regularly or irregularly spaced. Each point in the virtual space has one or more associated "view" or "current position" vectors which represent the view an operator would have when located at that point and when oriented to face a particular direction.

At each point in the virtual space, an operator navigating through the space may have a number of navigational options, or "degrees of freedom". For example, from any given point the operator may have the option to move forward, move backward, rotate left, rotate right, move up, or move down.

For each option at each point in the coordinate system, a motion vector is stored in motion vector database 14. The motion vectors represent the virtual path followed

by an operator when moving or translating from point to point, or when rotating about a fixed point in the virtual space. A separate motion vector is generated for each allowed movement (that is, translation or rotation) within the virtual space. The process by which motion vectors are generated, and the process by which the associated video sequences are rendered and encoded will be explained later.

Referring now to **Figure 3**, a sample reference grid **50** representing a rectangular coordinate system within a virtual space is shown. Grid **50** is a rectilinear grid extending in the X and Y directions as shown in **Figure 3**. Grid **50** may also extend into the Z direction, which is normal to the X-Y plane. The virtual space included in grid **50** may include a number of massing objects **52** which may, for example, represent stationary objects within a room. As discussed above, each point in grid **50** may have one or more associated "view" or "current position" vectors **54** which represent the view an operator would have when positioned at a particular point, and when looking in a particular direction. Each view or current position vector **54** has an associated field-of-view or frame-of-view **56**, which represents the extent of vision that an operator can have in any one direction, and describes the extent of peripheral vision available.

As can be seen in **Figure 4**, each node point P(x,y) in reference grid **50** preferably has associated with it several possible motion vectors **60** which, as discussed above, represent a translation to a different point in the virtual

space, or a rotation about a fixed point in the virtual space. Each motion vector may be a pair of vectors, one in the main direction and one in the return direction. There are preferably vectors for rotate right, rotate left, move forward, look up, and look down. Reciprocal vectors can be used to add motion options not specifically identified. For example, the motion backwards from the node point P(x,y) is the reciprocal of the motion forward to the point P(x-dx, y-dy), where dx and dy are incremental, arbitrarily chosen units of distance in the x and y directions, respectively. Alternatively, separate motion vectors for each possible translation and rotation (including reciprocal motions) could be generated and stored in motion vector database **14**.

The method by which a complete set of motion vectors can be created and stored in a motion vector database will now be described. Since motion vectors represent the virtual path followed by an operator when translating from point to point, or when rotating about a fixed point in the virtual space, a separate motion vector is generated for each allowed translation or rotation within the virtual space. Thus, the size of the motion vector database is proportional to the number of points in the virtual space, and the number of degrees of freedom an operator has when navigating through the virtual space. However, in many cases it will be possible to simplify the task of generating a database of motion vectors by constraining movement to certain portions of the virtual space. For example, most virtual spaces will have areas in which an operator will not need or expect to be

able to go, such as through solid objects, or to the underside of a table. By prohibiting movement in such areas, the total number of motion vectors required can be reduced.

Once it has been decided which areas in the virtual space can be traversed by the operator, a set of motion vectors can be calculated for each point in those areas. In a preferred embodiment, the motion vectors are generated automatically by development system 10 (See **Figure 2**). Appendix A includes a preferred computer program which is used to generate a motion vector database. As can be seen in Appendix A, routines are used to calculate all allowed translation and rotation paths from all allowed points in the virtual space.

After the translation and rotation paths represented by each motion vector have been stored in the database, the video sequences associated with those motion vectors are rendered, encoded, and stored in the database. The rendering is preferably performed using the same rendering package used to create the virtual space. Since rendering takes place "off-line" (that is, before an operator can navigate through the virtual space), the speed of rendering is irrelevant. Accordingly, extremely detailed images (which may take considerable time to render) can be used to show portions of or scenes within the virtual space.

In a preferred embodiment, system 10 automatically generates instructions used by the rendering package to dictate the position and movement of a "virtual camera" through the virtual space. Again, an appropriate rendering

package may be installed in and run by system **10**, or the rendering package may be installed in and run by a separate computer system. The use of virtual cameras in three dimensional rendering packages is known to those skilled in the art.

The virtual camera instructions, called "camera path data", indicate to the rendering package which view points and video sequences (associated with the previously defined motion vectors) should be rendered. As or after a video sequence has been rendered, the video sequence is encoded by video encoder **18** (see **Figure 2**) and stored in the motion vector database **14** linked to its associated motion vector.

The computer program included in Appendix A also contains routines which are used to generate camera path data. **Figure 5** is an example of camera path data as generated according to a preferred embodiment. The exemplary camera path data of **Figure 5** shows the movement instructions for a virtual camera (called "DCAM2") in the Y direction only, with no motion in the X and Z directions. Similar camera path data can be generated to record movement with components along the X and Z axes, and to represent rotation about a fixed point. Of course, any motion in any direction could be included in the camera path data.

The first row of **Figure 5** shows that the virtual camera DCAM2 begins at an X position of 156, a Y position of 24, and a Z position of 72. Recording of the video sequence begins at Frame 0. As can be seen in the subsequent rows,

DCAM2 moves in the Y direction in increments of 24 units for every 15 frames recorded. So in this example, the video sequence associated with a motion vector representing translation in the Y direction (from one point to another) would be 15 frames long, and would cover a distance of 24 units in the Y direction. The number of units moved in any direction depends upon the spacing (that is, distance) between points within the virtual space.

In an alternative embodiment, where the virtual space to be navigated through is based upon a real physical space, the camera path data could be used to control real automated cameras (sometimes called "motion controlled cameras") which move through and record images of the real space.

As discussed above, the video sequences may be encoded (using video encoder **18**) either while they are being rendered, or after they have been rendered. Each video sequence begins with a "key frame", which is a frame from which the video sequence can be accessed after it has been stored in the database. Each video sequence preferably also ends with a key frame which is the first frame of a possible subsequent video sequence.

In a preferred embodiment which incorporates MPEG encoding, each video sequence associated with a motion vector includes an MPEG Group of Pictures (GOP) which begins with an I frame (See **Figure 1**). Thus, in an embodiment incorporating MPEG encoding, the "key frames" correspond to the MPEG I frames. Each encoded video sequence associated with a motion

vector also ends with an I frame from a next possible video sequence. By always starting and ending an encoded video stream with an I frame (or other key frame), the appearance of smooth motion can be achieved when navigating through the virtual space. Again, I frames are typically placed in every 8 to 15 frames (i.e., every 1/4 to 1/2 second) of video footage. However, I frames could be placed closer together if dictated by the location or spacing of reference points within the virtual space, or by other factors.

After the motion vector database (including the associated encoded video sequences) is complete, a state machine is automatically created. This state machine keeps track of the position, orientation, and motion of an operator navigating through the virtual space; and based upon the position, orientation and motion of the operator in the virtual space, the state machine dictates which motion vector (more specifically, which video sequence associated with a particular motion vector) should be displayed at any given time. The state machine reflects the number of degrees of freedom an operator may have, and the navigational characteristics of the navigational controller **16** (see **Figures 2** and **8**) used to navigate through the virtual space. The use of state machines is well known in the art, and there are many different types of state machines that could be used with the present invention. Accordingly, the specific techniques used to generate a preferred state machine will not be described in detail.

Figures 6A - 6D contain graphical representations of selected motion vector node geometries according to a preferred embodiment. **Figures 6A - 6D** represent motion and orientation in two dimensions (here, the X-Y plane) only. However, it will be understood that similar figures could be used to represent motion and orientation in more than two dimensions.

Figure 6A represents the possible orientations and motions of an operator when facing or moving away from a node point in the virtual space. Each labeled arrow (A_n through A_{n+7}) represents a possible orientation (or direction of motion) when an operator is positioned at a particular node point P(x,y) in the virtual space. By way of analogy, each of the labeled arrows could be equated with the directions shown on a compass. For example, A_n could represent due north, A_{n+4} could represent due south, A_{n+5} could represent southwest, etc. Similarly, **Figure 6B** represents the possible orientations or directions of motion when moving into a particular node point P(x,y).

Figure 6C represents the motion of an operator when rotating to the right about a particular node point P(x,y). Here, the various motions (labeled C_n through C_{n+7}) represent one-eighth turns around the node point P(x,y). For example, C_n represents a one-eighth turn around a node point beginning facing in the "due north" direction, and ending facing in the "northeast" direction. Similarly, **Figure 6D** represents a left rotation about a fixed node point P(x,y). For example, the motion labeled D_{n+1} represents a one-eighth turn about a

node point beginning facing in the "due east" direction, and ending facing in the "northeast" direction.

Figures 7A - 7D are exemplary state tables which show beginning and end states for particular motions in the virtual space. Here, only the move forward, move backward, rotate left, and rotate right substates are shown. However, it will be understood that other substates, such as move up, move down, look up, and look down could also be included in the present invention.

The state tables shown in **Figures 7A - 7D** each include four columns. The first column includes the CPV, or current position vector. The current position vector shows the orientation of an operator at a particular point P(x,y) in the virtual space. For example, the first entry in the CPV column of **Figure 7A** shows an orientation in the A_{n+2} direction (see also **Figure 6A**).

The second column includes the MV (motion vector) representing a rotation about, or translation from, that point P(x,y). The third column includes the NPV, or next position vector. The next position vector represents the orientation of an operator after the rotation or translation represented by the motion vector has taken place. In **Figure 7A**, the NPV is always the current position vector since after moving backwards, the operator remains oriented in the same direction. The fourth column includes the NPCC, or next position change coordinate. The next position change coordinate here represents the change in the position of the operator when making a particular movement in the virtual

space. The NPCC code, as shown in **Figures 7A - 7D**, represents the type and direction of motion in the x and y directions caused by the execution of a motion vector. The normalized stepwise factor k represents an arbitrary unit of distance between points in the coordinate system, where x is defined as positive going to the right, and y is defined as positive going up. The operation of the state machine will be explained further in connection with the discussion of navigation through the virtual space.

NAVIGATION THROUGH THE VIRTUAL SPACE

After the motion vector database has been completed and the desired state machine established, a player program or "shell" can be used to play the video sequences associated with the stored motion vectors as the operator navigates through the virtual space. A preferred player program is included in Appendix A.

The player program preferably runs on a playback system **10A** as shown in **Figure 8**. System **10A** is nearly identical to the system **10** shown in **Figure 2**, except that a video encoder is not required in system **10A**. Also, the processor unit **12** in system **10A** (which again, may be a general purpose computer) need not be able to run or interface with a rendering package.

The player program operates the state machine discussed above to obtain the appropriate motion vector based on the current operator position, orientation, and motion. So when an operator navigating through the virtual space

changes his or her position by means of navigational controller **16**, the appropriate motion vector is selected from database **14**, and the encoded video sequence associated with it is decoded by decoder **20**, and displayed on display device **22**.

To produce a smooth transition from one video sequence to another, a preferred embodiment includes the concept of "virtual projectors". Such virtual projectors, shown conceptually in **Figure 9**, are used in the caching, queuing, and running of video sequences to be displayed on display device **22**.

Figure 11 is a block diagram showing a virtual projector system **80** according to a preferred embodiment. Virtual projector system **80** may be implemented in software or hardware to manage the caching, queuing, and running of video sequences. Virtual projector system **80** preferably includes a virtual projector control interface **82**, one or more source control elements **84**, and one or more surface control elements **86**. Virtual projector control interface **82** interacts with the state machine operated by the player program to determine which motion vectors (or more specifically, which associated video sequences) should be extracted from database **14**, and which extracted video sequences should actually be displayed on display device **22**. Virtual projector control interface **82** controls source control elements **84**, which are used to extract from motion vector database **14** the desired video sequences. Surface control elements **86**, which correspond to the individual projectors shown in **Figure 9**, are also

controlled by virtual projector control interface **82** and receive video sequence information from source control elements **84**. More specifically, virtual projector control interface **82** controls the timing and output of surface control elements **86**.

The outputs of surface control elements **86** are "displayed" on logical display surface **88**, which may include one or more virtual surfaces **88a** through **88n**. Surface control elements **86** may generate an output which is displayed on more than one virtual surface, and more than one surface control element **86** may project onto the same surface. Each surface **88a - 88n** may be assigned a specific priority of display. For example, the images displayed on surface **88a** may be made to appear in front of images displayed on surface **88c**. Such display techniques are commonly used, for example, in personal computers running a Microsoft Windows or Apple MacIntosh type user interface. The number of surfaces used is dependent upon the capabilities of the hardware and operation environment of playback system **10A** (see **Figure 8**). Using techniques known to those skilled in the art, the images displayed on surfaces **88a - 88n** may then be fed to and shown on display device **22** so as to be visible to the operator.

To seamlessly concatenate (or "string together") a series of video sequences, surface control elements **86** preferably function in a way similar to that of synchronized movie theater projectors. In a movie stored on several separate reels of film, the last few frames of film on one

reel are typically duplicated on a subsequent reel. By synchronizing the projectors, these identical frames can be projected simultaneously from both reels. When the film in one projector runs out, the second projector continues, creating a seamless transition between film reels.

The preferred virtual projector system **80** similarly uses the concepts of synchronization and frame overlap to allow a smooth transition between concatenated video sequences. As was discussed above, each video sequence associated with a motion vector begins with a "keyframe", which is an MPEG I frame in a preferred embodiment. Each video sequence also ends with a keyframe (preferably an MPEG I frame) that is identical to the first frame in a next possible video sequence. Thus, when a first surface control element **86** displays the last frame in a video sequence on a particular surface (for example, surface **88a**), a second surface control element **86** simultaneously displays the identical first frame in a subsequent sequence on that same surface. The second surface control element **86** then continues to display the remaining frames in that subsequent video sequence. This process is repeated for each video sequence displayed. Again, virtual projector control interface **82** controls the timing and output of each surface control element **86**. Of course, if the operator ceases to move through the virtual space, then the last frame of the last video sequence may be continuously displayed on display device **22**.

In addition to permitting the seamless concatenation of video sequences, independently controllable surface control elements **86** could also be used to provide picture-in-picture effects. For example, one surface control element could be used to display the scene outside a car's windshield, and another surface control element could be used to display the scene shown in a car's rear view mirror.

In addition, short MPEG video sequences (called Dynamic Motion Kernels, or DMKs) representing kinetic segments or effects within a larger scene (such as the motion of an object across a room) can be stored and indexed in the motion vector database **14**, and displayed using virtual projector system **80**.

FURTHER DETAILS OF STATE MACHINE OPERATION

As was discussed above with respect to **Figures 6A - 6D** and **7A - 7D**, a preferred embodiment incorporates a player program running a state machine to permit navigation through a virtual space. These Figures will be used explain below further details of state machine operation in a preferred embodiment, and the manner in which motion vectors are retrieved and queued.

When navigating through a virtual space, an operator is always at some point P(x,y) which may have several possible motion vectors associated with it. The state machine is used to keep track of which particular motion vector should be accessed next. For example, **Figure 7A** represents the state table for the "move backward"

substate. When an operator is at a node point P(x,y) facing the A_{n+2} direction (see **Figure 6A**) and attempts to move backward (for example, by pulling back on a joystick), the motion vector retrieved will be a B_{n+2} type motion vector (see **Figure 6B**). The video sequence associated with that particular motion vector will be queued and played by a virtual projector. The next position vector will be the previous current position vector (which is A_{n+2}), and the next position change coordinate will be x-k,y. That is, the operator begins the move at a particular point P(x,y) facing in the A_{n+2} direction, which is directly along the positive X direction. The operator moves backwards while facing the A_{n+2} direction, and thus ends at a point P(x-k,y) still facing the A_{n+2} direction. The B_{n+2} type motion vector representing that particular backwards move is retrieved, and the associated video segment displayed.

The state table for the "move forward" substate shown in **Figure 7B** is similar. For example, when an operator is at a node point P(x,y) facing the A_{n+1} direction and attempts to move forward (for example, by pushing forward on a joystick), the motion vector retrieved will be an A_{n+1} type motion vector. The video sequence associated with that particular motion vector will be queued and played by a virtual projector. The next position vector (i.e., the position vector when the next point is reached) remains A_{n+1}, and the next position change coordinate is x+k, y+k. That is, the operator begins at a node point P(x,y) facing in the A_{n+1} direction, moves in the A_{n+1} direction, and ends up at a node

point P(x+k, y+k) still facing in the A_{n+1} direction. The A_{n+1} type motion vector associated with that particular forward move is retrieved, and the associated video segment displayed.

The state tables for rotation about a fixed point P(x,y) are shown in **Figures 7C** and **7D**. For example, the first line of the table shown in **Figure 7C** specifies the turn right substate where an operator is at a point P(x,y) facing the A_n direction. If an operator attempts to rotate right about that point (for example, by pushing a joystick to the right), the motion vector retrieved will be a C_n type motion vector which represents a one-eighth rotation to the right. The video sequence associated with that particular motion vector will be queued and displayed. The next position vector is A_{n+1}, and the next position change coordinate is x,y (which is represented by the code 0). That is, an operator begins at a point P(x,y) facing the A_n direction. The operator then makes a one-eighth turn around the point and remains at point P(x,y), but now faces the A_{n+1} direction. The C_n type motion vector representing that particular rotation is retrieved, and the associated video segment displayed.

The operation of the state table representing a left rotation is analogous to the operation of the right rotation state table, and will not be discussed here.

EFFICIENT RETRIEVAL OF MOTION VECTORS

The motion vectors are preferably arranged in database **14** with a specific schema designed to optimize the relationship between the motion vectors' geometrical relationship in the virtual space, and the representation of those vectors in the layout of the database (which is preferably on a recorded magnetic and/or optical disk). For example, the overall speed and responsiveness of the system could be enhanced by locating the data for motion vectors likely to be played close together in time physically close together on the disk or other storage medium. Other retrieval time optimization techniques know to those skilled in the art could also be used.

To further increase the speed of motion vector retrieval, a set of motion vectors can also be characterized mathematically as a set of "probability vectors" (see **Figure 10**). Taken as a whole, this set is equivalent to a "probability envelope". That is, at any given point P(x,y,z) in the virtual space there may be several possible motion vectors (shown in **Figure 10** as Mv0 - Mvm) representing possible translations or rotations in the virtual space. The probability that motion vectors associated with that point (or with nearby points) will be required soon is relatively high. The probability that a motion vector associated with a more distant point will be needed soon is lower. So the probability that a particular vector will be used to generate the next video sequence displayed is determined by the

position and direction of the operator traveling through the virtual space, and by any rules that may apply.

Computer programs can be used to include any particular factors in determining motion vector probabilities. The database engine can also be optimized for particular constraints or probability-determining factors. The factors used in determining which motion vector in the envelope is most likely to be needed next can include position, direction, velocity, and higher order factors.

By using virtual projector system **80** (see **Figure 11**) to cache and queue the next possible motion vectors according to their probability of being called (such that the ones most likely to be called are queued most rapidly for display), the overall speed and responsiveness of the system can be enhanced. For example, a point P(x,y,z) within the virtual space may have five possible motion vectors associated with it. So when an operator arrives at or near the point P(x,y,z), the five video sequences associated with those five possible motion vectors could be cached and queued in five surface control elements **86**. After the operator indicates (through navigational controller **16** (see **Figure 8**)) which motion vector is appropriate, the virtual projector control interface **82** commands the surface control element **86** containing the video sequence associated with that motion vector to display the sequence in the manner discussed above. Unplayed video sequences cached in surface control elements **86** are simply discarded, ignored, or erased.

Object Constraint Systems

Each scene observed by the operator of a virtual reality or interactive computer system has three kinds of objects: stationary scene objects which form the background, constrained objects that exhibit a limited ability to move against the background, and free-floating or unconstrained objects. In most cases, the largest number of objects in a given scene are of the stationary category, followed next by constrained and then free-floating objects. Regardless of the type of object, current virtual reality or interactive computer systems require each object to be fully rendered for each frame in which it is displayed. The present invention reduces the number of calculations required to present scenes. However, the number of motion vectors available may be limited by the size of the motion vector database. Accordingly, storing a large number of scenes including different possible positions for free-floating objects may not be practical.

The virtual projector of the preferred embodiment can also be used advantageously in this situation. In a preferred embodiment, free-floating objects (which are rendered in real time) can be projected on a second video display plane, which would overlay the video planes of the stationary and constrained motion prerendered objects. This allows unconstrained manipulation of free-floating objects using standard input or operator interface devices. And since free-floating objects tend to be small, the rendering

of these objects in real time can be performed with fairly high resolution.

Representing Velocity Changes

Changes in an operator's apparent translational or rotational velocity in the virtual space can be controlled through the use of a variable speed video decoder **20** (see **Figure 8**). If video decoder **20** is not a variable speed video decoder, realistic motion vector simulation of acceleration and deceleration can be achieved by duplication of frame data, and by frame "jumping". That is, the more duplicated frames there are in a video sequence, the slower the operator's apparent velocity. Conversely, by skipping frames in a video sequence, the operator's apparent velocity through the virtual space can be increased.

The present invention has been described in terms of a preferred embodiment. However, the scope of the invention is defined by the appended claims.

Appendix A

U.S. Patent Application Serial No.: Not yet assigned
Filing Date: February 14, 1996 (filed concurrently herewith)
For: METHOD AND SYSTEM FOR THE CREATION OF AND NAVIGATION THROUGH A MULTIDIMENSIONAL SPACE USING ENCODED DIGITAL VIDEO
By: Gerald M. Segal
Atty. Docket No.: 8680-003-999

```
REM
===========================================================
// All Variables are defined ithe is module for the Generation programs
///
===========================================================
Global X1 As Integer
Global X2 As Integer
Global X3 As Integer
Global X4 As Integer
Global Y1 As Integer
Global Y2 As Integer
Global Y3 As Integer
Global Y4 As Integer
Global Z1 As Integer
Global Z2 As Integer
Global KeyScale As Integer
Global FrameGroup As Integer
Global ObjectName As String
Global FileNameDir As String
Global XAsend As Integer
Global YAsend As Integer
Global XReturn As Integer
Global YReturn As Integer
Global PitchFlag As Integer
Global RevMode As Integer
Global RollFlag As Integer
Global YawFlag As Integer
Global SkewFlag As Integer
Global AutoFileMode As Integer
Global ZSingleSlopeFunction As Integer
Global ZStepFunction As Integer
Global ZConstantFunction As Integer
Global ZEValue As Integer
Global ZFValue As Integer
Global ZAValue As Integer
Global ZBValue As Integer
Global RootFN As String
Global SeqNum As Integer
Global KeyNum As Integer
Global MPGIndex As Integer
Global QuadNum As Integer
'----------------------------------

Function GetMPGFilename () // This Gets the next filename of a motion vector no dups.
   MPGIndex = MPGIndex + 1
   KeySeq$ = Format$(MPGIndex, "0000")
   GetMPGFilename = RootFN + KeySeq$ + ".MPG"
End Function

Function GetNextFileString () // Get directory string
```

```
        SeqNum = SeqNum + 1
         Seq$ = Format$(SeqNum, "0000")
        GetNextFileString = FileNameDir + RootFN + Seq$ + ".ASC"
End Function

Sub RotateMultiY () // calculates a series of right rotational motion vectors
    RootFN = "ROTR"
    '---------- db ----------
    Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
    Form3.Data1.RecordSource = "Quadrant"
    Form3.Data1.Refresh
    If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
    End If
    '---------- end db ----------
    For YValue = Y1 To Y2 Step KeyScale
    XValue = X2
    ZF1Value = (YValue - Y1) / KeyScale
    If ZStepFunction = True Then
       ZValue = ZAValue + ZBValue
    End If
    If ZConstantFunction = True Then
       ZValue = Z1
    End If
    If ZSingleSlopeFunction = True Then
       ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
    End If
    Fnumb = 0
    FnumbEnd = FrameGroup * 8
    AngleStart = 0
    AngleEnd = 360
    '---------------------------------------
    Do Until XValue > X3
      RVector = 21
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      'Debug.Print Header$
      OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
      Print #1, OutStrNext$
      'Debug.Print OutStrNext$
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "0, 0, 1, " + Str$(AngleStart)
+ ", " + Str$(Fnumb)
      Print #1, OutStrRotate$
      'Debug.Print OutStrRotate$
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "0, 0, 1, " + Str$(AngleEnd) +
", " + Str$(FnumbEnd)
      Print #1, OutStrRotate$
      'Debug.Print OutStrRotate$
      OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " + Str$(Fnumb)
```

```
      Print #1, OutStrCBTF$
      Fnumb1 = Fnumb
      For N = 1 To 8
      '--------------- db -----------------------------
         Form3.Data1.Recordset.AddNew
         Form3.Data1.Recordset("KeyNo") = KeyNum
         Form3.Data1.Recordset("Xcoord") = XValue
         Form3.Data1.Recordset("Ycoord") = YValue
         Form3.Data1.Recordset("Zcoord") = ZValue
         Form3.Data1.Recordset("Direction_Vector") = RVector
         Form3.Data1.Recordset("QuadNo") = QuadNum
         Form3.Data1.Recordset("Frame#") = Fnumb1
         Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
         ST1$ = Right$(ASCFileName$, 12)
         ST2$ = Left$(ST1$, 8) + ".VUE"
         Form3.Data1.Recordset("VUE_Filename") = ST2$
         Mpgfilename$ = GetMPGFilename()
         Form3.Data1.Recordset("Stream_File") = Mpgfilename$
         Form3.Data1.Recordset.Update
         RVector = RVector + 2
         Fnumb1 = Fnumb1 + FrameGroup
         KeyNum = KeyNum + 1
         '------------------------------ end db -----------------------------
      Next N
      XValue = XValue + KeyScale
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      Print #1, OutStrScale$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
   Loop
   Next YValue
   '---------------- db ------------------
      Form3.Data1.Recordset.Close
   '------------------ db end ------------
End Sub
'// ---------------------------------------------------------------------------------------------------------
Sub RotateMultiYB () '// Calculates a series of Left rotational motion vectors
   RootFN = "ROTL"
   '---------------- db ----------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '----------------- end db ----------------
   For YValue = Y1 To Y2 Step KeyScale
   XValue = X2
   ZF1Value = (YValue - Y1) / KeyScale
   If ZStepFunction = True Then
      ZValue = ZAValue + ZBValue
```

```
    End If
    If ZConstantFunction = True Then
       ZValue = Z1
    End If
    If ZSingleSlopeFunction = True Then
       ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
    End If
    Fnumb = 0
    FnumbEnd = FrameGroup * 8
    AngleStart = 0
    AngleEnd = 360
    '---------------------------------------
     Do Until XValue > X3
       RVector = 36
       ASCFileName$ = GetNextFileString()
       Fnumb1 = Fnumb
       For N = 1 To 8
       '--------------- db -----------------------------
          Form3.Data1.Recordset.AddNew
          Form3.Data1.Recordset("KeyNo") = KeyNum
          Form3.Data1.Recordset("Xcoord") = XValue
          Form3.Data1.Recordset("Ycoord") = YValue
          Form3.Data1.Recordset("Zcoord") = ZValue
          Form3.Data1.Recordset("Direction_Vector") = RVector
          Form3.Data1.Recordset("QuadNo") = QuadNum
          Form3.Data1.Recordset("Frame#") = Fnumb1
          Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
          ST1$ = Right$(ASCFileName$, 12)
          ST2$ = Left$(ST1$, 8) + ".VUE"
          Form3.Data1.Recordset("VUE_Filename") = ST2$
          Mpgfilename$ = GetMPGFilename()
          Form3.Data1.Recordset("Stream_File") = Mpgfilename$
          Form3.Data1.Recordset.Update
          RVector = RVector - 2
          Fnumb1 = Fnumb1 + FrameGroup
          KeyNum = KeyNum + 1
          '------------------------------ end db -----------------------------
       Next N
       XValue = XValue + KeyScale
    Loop
    Next YValue
    '---------------- db ------------------
      Form3.Data1.Recordset.Close
    '------------------- db end ------------
End Sub

Sub SkewNEMulti () ' // builds NE oriented Vectors = =
   RootFN = "SKNE"
   DY = Y2 - Y1
   DX = X3 - X2
   Fnumb = 0
   If ZStepFunction = True Then
```

```
      ZValue = ZAValue + ZBValue
    End If
    If ZConstantFunction = True Then
      ZValue = Z1
    End If
    '----------------- db -----------------
    Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
    Form3.Data1.RecordSource = "Quadrant"
    Form3.Data1.Refresh
    If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
    End If
    '----------------- end db --------------------
     For N = 1 To (DX / KeyScale)
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      'Debug.Print Header$
      Fnumb = 0
      XValue = X1
      YValue = Y2 - (N * KeyScale)
      YLast = YValue
      ZF1Value = ZFValue - N
      Do Until (XValue > X1 + (N * KeyScale))
      '-------------------------------------------
        Form3.Data1.Recordset.AddNew
      '-------------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
        'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '---------------------------------- db ----------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 13
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
```

```
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db -----------------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue + KeyScale
        ZF1Value = ZF1Value + 1
      Loop
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
    Next N
    '--------------------- 1st 3rd -------------------
    ZF1Value = ZEValue - 1
    NS = (DY - DX) / KeyScale
    For N = 1 To NS
    YValue = YLast - (N * KeyScale)
    XValue = X1
    ASCFileName$ = GetNextFileString()
    Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
    Open ASCFileName$ For Append As #1
    Print #1, Header$
    'Debug.Print Header$
    Fnumb = 0
      Do Until XValue > X3
      '-----------------------------------------
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
        'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '------------------------- db -------------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 13
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
```

```
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue + KeyScale
        ZF1Value = ZF1Value + 1
      Loop
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
      ZF1Value = ZEValue - 1
    Next N
    '------------------ Next 3rd ------------------
    YValue = Y1
    YEndValue = Y3 - KeyScale
    For N = 1 To (DX / KeyScale) - 1
      XValue = X1 + (N * KeyScale)
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      'Debug.Print Header$
      Fnumb = 0
      Do Until XValue > X4
      '-------------------------------------------
        Form3.Data1.Recordset.AddNew
      '-------------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
        Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '------------------------- db --------------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 13
```

```
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue + KeyScale
        ZF1Value = ZF1Value + 1
      Loop
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
      YValue = Y1
      ZF1Value = ZEValue - 1
      XValue = XValue + KeyScale
      Next N
      '----------------- db -------------------
      Form3.Data1.Recordset.Close
      '------------------- db end ------------
End Sub

Sub SkewNEMultiB () '// Builds NE oriented vectors with a backward direction
   RootFN = "NEBK"
   DY = Y2 - Y1
   DX = X3 - X2
   Fnumb = 0
   If ZStepFunction = True Then
      ZValue = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '----------------- db -----------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
     Form3.Data1.Recordset.MoveLast
   End If
   '------------------ end db -------------------
    For N = 1 To (DX / KeyScale)
```

```
    ASCFileName$ = GetNextFileString()
    Fnumb = 0
    XValue = X1 + (N * KeyScale)
    YValue = Y2
    ZF1Value = ZFValue
    Do Until (XValue < X1)
    '-------------------------------------------
      Form3.Data1.Recordset.AddNew
    '-------------------------------------------
      If ZSingleSlopeFunction = True Then
        ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
      End If
      '---------------------------------- db ----------------------
      Form3.Data1.Recordset("KeyNo") = KeyNum
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 14
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1
      Form3.Data1.Recordset.Update
      '------------------------------ end db ----------------------------
      XValue = XValue - KeyScale
      Fnumb = FrameGroup + Fnumb
      YValue = YValue - KeyScale
      ZF1Value = ZF1Value - 1
    Loop
Next N
ZF1Value = ZFValue
NS = (DY - DX) / KeyScale
For N = 1 To NS
    XValue = X3
    YValue = Y3 - (N * KeyScale)
    YLast = YValue
    ASCFileName$ = GetNextFileString()
    Fnumb = 0
    Do Until XValue < X1
    '-------------------------------------------
      Form3.Data1.Recordset.AddNew
    '-------------------------------------------
      If ZSingleSlopeFunction = True Then
        ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
      End If
      '---------------------------------- db ----------------------
      Form3.Data1.Recordset("KeyNo") = KeyNum
```

```
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 14
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1
      Form3.Data1.Recordset.Update
      '------------------------------ end db ----------------------------
      XValue = XValue - KeyScale
      Fnumb = FrameGroup + Fnumb
      YValue = YValue - KeyScale
      ZF1Value = ZF1Value - 1
   Loop
   ZF1Value = ZFValue
 Next N
 XValue = X4
For N = 1 To (DX / KeyScale) - 1
YValue = YLast - (N * KeyScale)
ZF1Value = ZFValue - 1
   ASCFileName$ = GetNextFileString()
   Fnumb = 0
   Do Until YValue < Y1
   '-----------------------------------------------
     Form3.Data1.Recordset.AddNew
   '-----------------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
      End If
      '---------------------------------- db ----------------------
      Form3.Data1.Recordset("KeyNo") = KeyNum
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 14
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1
      Form3.Data1.Recordset.Update
      '------------------------------ end db ----------------------------
```

```
        XValue = XValue - KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
      Loop
      ZF1Value = ZFValue - (N + 1)
      XValue = X4
      Next N
      '---------- db ----------
      Form3.Data1.Recordset.Close
      '---------- db end ----------
End Sub

Sub SkewNWMulti () '\\-------builds NW motionvectors forward.
   RootFN = "SKNW"
   DY = Y2 - Y1
   DX = X4 - X1
   Fnumb = 0
   If ZStepFunction = True Then
      ZValue = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '---------- db ----------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '---------- end db ----------
   For N = 1 To (DX / KeyScale)
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      'Debug.Print Header$
      Fnumb = 0
      XValue = X1 + (N * KeyScale)
      YValue = Y1
      ZF1Value = ZEValue - 1
      Do Until (XValue < X1)
      '-----------------------------------------
       Form3.Data1.Recordset.AddNew
      '-----------------------------------------
         If ZSingleSlopeFunction = True Then
            ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
         End If
         OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
         OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
```

```
Str$(Fnumb)
        Print #1, OutStrNext$
        Print #1, OutStrCBTF$
        'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$

        '---------------------------------- db ----------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 19
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------------
        XValue = XValue - KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue + KeyScale
        ZF1Value = ZF1Value + 1
    Loop
    OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
    OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
    Print #1, OutStrScale$
    Print #1, OutStrRotate$
    Print #1, "End"
    'Debug.Print "End"
    Print #1, OutStrCBTF$
    Close #1
  Next N
  NS = DX / KeyScale
  For N = 1 To NS
     XValue = X4
    ASCFileName$ = GetNextFileString()
    Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
    Open ASCFileName$ For Append As #1
    Print #1, Header$
    'Debug.Print Header$
    ZF1Value = ZEValue - 1
    YValue = (N * KeyScale) + KeyScale
    YLast = YValue
    Fnumb = 0
    Do Until XValue < X1
    '-----------------------------------------
      Form3.Data1.Recordset.AddNew
```

```
        '-----------------------------------------
            If ZSingleSlopeFunction = True Then
               ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
            End If
            OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
            OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
            Print #1, OutStrNext$
            Print #1, OutStrCBTF$
            'Debug.Print OutStrNext$
            '---------------------------------- db ----------------------
            Form3.Data1.Recordset("KeyNo") = KeyNum
            Form3.Data1.Recordset("Xcoord") = XValue
            Form3.Data1.Recordset("Ycoord") = YValue
            Form3.Data1.Recordset("Zcoord") = ZValue
            Form3.Data1.Recordset("Direction_Vector") = 19
            Form3.Data1.Recordset("QuadNo") = QuadNum
            Form3.Data1.Recordset("Frame#") = Fnumb
            Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
            ST1$ = Right$(ASCFileName$, 12)
            ST2$ = Left$(ST1$, 8) + ".VUE"
            Form3.Data1.Recordset("VUE_Filename") = ST2$
            Mpgfilename$ = GetMPGFilename()
            Form3.Data1.Recordset("Stream_File") = Mpgfilename$
            KeyNum = KeyNum + 1
            Form3.Data1.Recordset.Update
            '------------------------------ end db -----------------------------
            XValue = XValue - KeyScale
            Fnumb = FrameGroup + Fnumb
            YValue = YValue + KeyScale
            ZF1Value = ZF1Value + 1
        Loop
        OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
        OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
        Print #1, OutStrScale$
        Print #1, OutStrRotate$
        Print #1, "End"
        'Debug.Print "End"
        Close #1
        ZF1Value = ZF1Value + 1
    Next N
    NYs = ((DX - KeyScale) / KeyScale) + 1
    YValue = YLast + KeyScale
    XValue = X4
    ZF1Value = ZEValue
    For N = 1 To NYs
        ASCFileName$ = GetNextFileString()
        Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
        Open ASCFileName$ For Append As #1
        Print #1, Header$
        'Debug.Print Header$
```

```
        Fnumb = 0
        Do Until YValue > Y3
        '----------------------------------------
          Form3.Data1.Recordset.AddNew
        '----------------------------------------
          If ZSingleSlopeFunction = True Then
            ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
          End If
          OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
          OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
          Print #1, OutStrNext$
          Print #1, OutStrCBTF$
          'Debug.Print OutStrNext$
          '---------------------------------- db ----------------------
          Form3.Data1.Recordset("KeyNo") = KeyNum
          Form3.Data1.Recordset("Xcoord") = XValue
          Form3.Data1.Recordset("Ycoord") = YValue
          Form3.Data1.Recordset("Zcoord") = ZValue
          Form3.Data1.Recordset("Direction_Vector") = 19
          Form3.Data1.Recordset("QuadNo") = QuadNum
          Form3.Data1.Recordset("Frame#") = Fnumb
          Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
          ST1$ = Right$(ASCFileName$, 12)
          ST2$ = Left$(ST1$, 8) + ".VUE"
          Form3.Data1.Recordset("VUE_Filename") = ST2$
          Mpgfilename$ = GetMPGFilename()
          Form3.Data1.Recordset("Stream_File") = Mpgfilename$
          KeyNum = KeyNum + 1
          Form3.Data1.Recordset.Update
          '------------------------------ end db -----------------------------
          XValue = XValue - KeyScale
          Fnumb = FrameGroup + Fnumb
          YValue = YValue + KeyScale
          ZF1Value = ZF1Value + 1
        Loop
        OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
        OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
        Print #1, OutStrScale$
        Print #1, OutStrRotate$
        Print #1, "End"
        'Debug.Print "End"
        Close #1
        YValue = YLast + (N * KeyScale) + KeyScale
        ZF1Value = ZEValue + N
        XValue = X4
        Next N
        '----------------- db -------------------
        Form3.Data1.Recordset.Close
        '------------------- db end ------------
End Sub
```

```
'| <------------------------------------------------------------------------------>
Sub SkewNWMultiB () ' \\Builds NW oriented Motion Vectors Backwards
   RootFN = "NWBK"
   DY = Y2 - Y1
   DX = X3 - X2
   Fnumb = 0
   If ZStepFunction = True Then
      ZValue = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '----------------- db -----------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '----------------- end db --------------------
    For N = 1 To (DX / KeyScale)  '--- 1 ---
      ASCFileName$ = GetNextFileString()
      Fnumb = 0
      XValue = X1
      YValue = Y1 + (N * KeyScale)
      LastY = YValue
      ZF1Value = ZEValue + (N - 1)
      Do Until YValue < Y1
      '-----------------------------------------------
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------------
         If ZSingleSlopeFunction = True Then
            ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
         End If
         '---------------------------------- db ------------------------
         Form3.Data1.Recordset("KeyNo") = KeyNum
         Form3.Data1.Recordset("Xcoord") = XValue
         Form3.Data1.Recordset("Ycoord") = YValue
         Form3.Data1.Recordset("Zcoord") = ZValue
         Form3.Data1.Recordset("Direction_Vector") = 20
         Form3.Data1.Recordset("QuadNo") = QuadNum
         Form3.Data1.Recordset("Frame#") = Fnumb
         Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
         ST1$ = Right$(ASCFileName$, 12)
         ST2$ = Left$(ST1$, 8) + ".VUE"
         Form3.Data1.Recordset("VUE_Filename") = ST2$
         Mpgfilename$ = GetMPGFilename()
         Form3.Data1.Recordset("Stream_File") = Mpgfilename$
         KeyNum = KeyNum + 1
         Form3.Data1.Recordset.Update
         '------------------------------ end db ----------------------------
         XValue = XValue + KeyScale
```

```
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
      Loop
    Next N
    NS = (DY - DX) / KeyScale
    For N = 1 To NS
      XValue = X1
      ZF1Value = ZFValue
      YValue = LastY + KeyScale
      Fnumb = 0
      ASCFileName$ = GetNextFileString()
      LastY = YValue
      Do Until XValue > X3
      '-----------------------------------------
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        '---------------------------------- db ----------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 20
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
      Loop
      ZF1Value = ZFValue
    Next N
  YValue = Y2
  For N = 1 To (DX / KeyScale) - 1
    XValue = X1 + (N * KeyScale)
    ZF1Value = ZFValue - 1
    Fnumb = 0
    ASCFileName$ = GetNextFileString()
    Do Until XValue > X4
    '-----------------------------------------
```

```
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        '---------------------------------- db ------------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 20
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db -----------------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
      Loop
      YValue = Y3
      ZF1Value = ZFValue - (N + 1)
      XValue = X4 - DY + ((N + 1) * KeyScale)
      Next N
      '----------------- db -------------------
      Form3.Data1.Recordset.Close
      '------------------- db end -------------
End Sub
'<-------------------------------------------------------------------------------------------->
Sub SkewSEMulti () '// = Builds SE Oriented Motion Vectors
  RootFN = "SKSE"
  DY = Y2 - Y1
  DX = X3 - X2
  Fnumb = 0
  If ZStepFunction = True Then
    ZValue = ZAValue + ZBValue
  End If
  If ZConstantFunction = True Then
    ZValue = Z1
  End If
  '----------------- db -----------------
  Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
  Form3.Data1.RecordSource = "Quadrant"
  Form3.Data1.Refresh
  If Form3.Data1.Recordset.BOF = False Then
```

```
      Form3.Data1.Recordset.MoveLast
    End If
    '------------------ end db -------------------
    For N = 1 To (DX / KeyScale)  '--- 1 ---
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      'Debug.Print Header$
      Fnumb = 0
      XValue = X1
      YValue = Y1 + (N * KeyScale)
      LastY = YValue
      ZF1Value = ZEValue + (N - 1)
      Do Until YValue < Y1
      '-----------------------------------------
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
        'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '---------------------------------- db ----------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 15
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
    Loop
    OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
    OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
```

```
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
    Next N
    NS = (DY - DX) / KeyScale
    For N = 1 To NS
      XValue = X1
      ZF1Value = ZFValue
      YValue = LastY + KeyScale
      Fnumb = 0
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      'Debug.Print Header$
      LastY = YValue
      Do Until XValue > X3
      '-----------------------------------------
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
        'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '---------------------------------- db ------------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 15
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------------
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
```

```
        ZF1Value = ZF1Value - 1
      Loop
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
      ZF1Value = ZFValue
    Next N
   YValue = Y2
   For N = 1 To (DX / KeyScale) - 1
     XValue = X1 + (N * KeyScale)
     ZF1Value = ZFValue - 1
     Fnumb = 0
     ASCFileName$ = GetNextFileString()
     Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
     Open ASCFileName$ For Append As #1
     Print #1, Header$
     'Debug.Print Header$
     Do Until XValue > X4
     '-----------------------------------------
       Form3.Data1.Recordset.AddNew
     '-----------------------------------------
       If ZSingleSlopeFunction = True Then
         ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
       End If
       OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
       OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
       Print #1, OutStrNext$
       'Debug.Print OutStrNext$
       Print #1, OutStrCBTF$
       '---------------------------------- db ----------------------
       Form3.Data1.Recordset("KeyNo") = KeyNum
       Form3.Data1.Recordset("Xcoord") = XValue
       Form3.Data1.Recordset("Ycoord") = YValue
       Form3.Data1.Recordset("Zcoord") = ZValue
       Form3.Data1.Recordset("Direction_Vector") = 15
       Form3.Data1.Recordset("QuadNo") = QuadNum
       Form3.Data1.Recordset("Frame#") = Fnumb
       Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
       ST1$ = Right$(ASCFileName$, 12)
       ST2$ = Left$(ST1$, 8) + ".VUE"
       Form3.Data1.Recordset("VUE_Filename") = ST2$
       Mpgfilename$ = GetMPGFilename()
       Form3.Data1.Recordset("Stream_File") = Mpgfilename$
       KeyNum = KeyNum + 1
       Form3.Data1.Recordset.Update
       '------------------------------ end db -----------------------------
```

```
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
      Loop
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
      YValue = Y3
      ZF1Value = ZFValue - (N + 1)
      XValue = X4 - DY + ((N + 1) * KeyScale)
      Next N
      '---------- db ----------
      Form3.Data1.Recordset.Close
      '---------- db end ----------
End Sub

Sub SkewSEMultiB () '// ------- Builds SE Motion Vectors Backwards
   'RootFN = "SEBK"
   'DY = Y2 - Y1
   'DX = X3 - X2
   'Fnumb = 0
   'If ZStepFunction = True Then
    '  ZValue = ZAValue + ZBValue
   'End If
   'If ZConstantFunction = True Then
   '   ZValue = Z1
   'End If
   '---------- db ----------
   'Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   'Form3.Data1.RecordSource = "Quadrant"
   'Form3.Data1.Refresh
   'If Form3.Data1.Recordset.BOF = False Then
    ' Form3.Data1.Recordset.MoveLast
   'End If
   '---------- end db ----------
   ' For N = 1 To (DX / KeyScale)
     ' ASCFileName$ = GetNextFileString()
     ' Fnumb = 0
     ' XValue = X1 + (N * KeyScale)
     ' YValue = Y2
     ' ZF1Value = ZFValue
      'Do Until (XValue < X1)
      '-----------------------------------------
      '  Form3.Data1.Recordset.AddNew
      '-----------------------------------------
      '   If ZSingleSlopeFunction = True Then
      '      ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
      '   End If
        '---------- db ----------
```

```
'    Form3.Data1.Recordset("KeyNo") = KeyNum
'    Form3.Data1.Recordset("Xcoord") = XValue
'    Form3.Data1.Recordset("Ycoord") = YValue
'    Form3.Data1.Recordset("Zcoord") = ZValue
'    Form3.Data1.Recordset("Direction_Vector") = 16
'    Form3.Data1.Recordset("QuadNo") = QuadNum
'    Form3.Data1.Recordset("Frame#") = Fnumb
'    Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
'    ST1$ = Right$(ASCFileName$, 12)
'    ST2$ = Left$(ST1$, 8) + ".VUE"
'    Form3.Data1.Recordset("VUE_Filename") = ST2$
'    Mpgfilename$ = GetMPGFilename()
'    Form3.Data1.Recordset("Stream_File") = Mpgfilename$
'    KeyNum = KeyNum + 1
'    Form3.Data1.Recordset.Update
     '------------------------------ end db ----------------------------
'    XValue = XValue - KeyScale
'    Fnumb = FrameGroup + Fnumb
'    YValue = YValue - KeyScale
'    ZF1Value = ZF1Value - 1
' Loop
'Next N
'ZF1Value = ZFValue
'NS = (DY - DX) / KeyScale
'For N = 1 To NS
 ' XValue = X3
 ' YValue = Y3 - (N * KeyScale)
 ' YLast = YValue
 ' ASCFileName$ = GetNextFileString()
 ' Fnumb = 0
 ' Do Until XValue < X1
  '-------------------------------------------
  '  Form3.Data1.Recordset.AddNew
  '-------------------------------------------
  '   If ZSingleSlopeFunction = True Then
  '      ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
  '   End If
     '---------------------------------- db ----------------------
'    Form3.Data1.Recordset("KeyNo") = KeyNum
'    Form3.Data1.Recordset("Xcoord") = XValue
'    Form3.Data1.Recordset("Ycoord") = YValue
'    Form3.Data1.Recordset("Zcoord") = ZValue
'    Form3.Data1.Recordset("Direction_Vector") = 16
'    Form3.Data1.Recordset("QuadNo") = QuadNum
'    Form3.Data1.Recordset("Frame#") = Fnumb
'    Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
'    ST1$ = Right$(ASCFileName$, 12)
'    ST2$ = Left$(ST1$, 8) + ".VUE"
'    Form3.Data1.Recordset("VUE_Filename") = ST2$
'    Mpgfilename$ = GetMPGFilename()
'    Form3.Data1.Recordset("Stream_File") = Mpgfilename$
'    KeyNum = KeyNum + 1
```

```
'     Form3.Data1.Recordset.Update
''------------------------------ end db ------------------------------
'     XValue = XValue - KeyScale
'     Fnumb = FrameGroup + Fnumb
'     YValue = YValue - KeyScale
'     ZF1Value = ZF1Value - 1
' Loop
' ZF1Value = ZFValue
'Next N
' XValue = X4
'For N = 1 To (DX / KeyScale) - 1
'YValue = YLast - (N * KeyScale)
'ZF1Value = ZFValue - 1
  'ASCFileName$ = GetNextFileString()
  'Fnumb = 0
  'Do Until YValue < Y1
  '-----------------------------------------
   ' Form3.Data1.Recordset.AddNew
  '-----------------------------------------
   ' If ZSingleSlopeFunction = True Then
   '    ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
   ' End If
   ' Form3.Data1.Recordset("KeyNo") = KeyNum
   ' Form3.Data1.Recordset("Xcoord") = XValue
   ' Form3.Data1.Recordset("Ycoord") = YValue
   ' Form3.Data1.Recordset("Zcoord") = ZValue
   ' Form3.Data1.Recordset("Direction_Vector") = 16
   ' Form3.Data1.Recordset("QuadNo") = QuadNum
   ' Form3.Data1.Recordset("Frame#") = Fnumb
   ' Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
   ' ST1$ = Right$(ASCFileName$, 12)
   ' ST2$ = Left$(ST1$, 8) + ".VUE"
   ' Form3.Data1.Recordset("VUE_Filename") = ST2$
   ' Mpgfilename$ = GetMPGFilename()
   ' Form3.Data1.Recordset("Stream_File") = Mpgfilename$
   ' KeyNum = KeyNum + 1
   ' Form3.Data1.Recordset.Update
    '------------------------------ end db ------------------------------
   ' XValue = XValue - KeyScale
   ' Fnumb = FrameGroup + Fnumb
   ' YValue = YValue - KeyScale
   ' ZF1Value = ZF1Value - 1
  'Loop
  'ZF1Value = ZFValue - (N + 1)
  'XValue = X4
  'Next N
  '---------------- db ------------------
  'Form3.Data1.Recordset.Close
  '------------------ db end -------------
  ' Begin New routine ----- sknw
RootFN = "SEBK"
DY = Y2 - Y1
```

```
DX = X4 - X1
Fnumb = 0
If ZStepFunction = True Then
   ZValue = ZAValue + ZBValue
End If
If ZConstantFunction = True Then
   ZValue = Z1
End If
'---------- db ----------
Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
Form3.Data1.RecordSource = "Quadrant"
Form3.Data1.Refresh
If Form3.Data1.Recordset.BOF = False Then
   Form3.Data1.Recordset.MoveLast
End If
'---------- end db ----------
For N = 1 To (DX / KeyScale)
   ASCFileName$ = GetNextFileString()
   '---------- del ----- here ----------
   Fnumb = 0
   XValue = X1 + (N * KeyScale)
   YValue = Y1
   ZF1Value = ZEValue - 1
   Do Until (XValue < X1)
   '-----------------------------------------
   Form3.Data1.Recordset.AddNew
   '-----------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
      End If
      '---------- db ----------
      Form3.Data1.Recordset("KeyNo") = KeyNum
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 16
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1
      Form3.Data1.Recordset.Update
      '---------- end db ----------
      XValue = XValue - KeyScale
      Fnumb = FrameGroup + Fnumb
      YValue = YValue + KeyScale
      ZF1Value = ZF1Value + 1
Loop
```

```
Next N
NS = DX / KeyScale
For N = 1 To NS
   XValue = X4
  ASCFileName$ = GetNextFileString()
  ZF1Value = ZEValue - 1
  YValue = (N * KeyScale) + KeyScale
  YLast = YValue
  Fnumb = 0
  Do Until XValue < X1
  '-------------------------------------------
    Form3.Data1.Recordset.AddNew
  '-------------------------------------------
     If ZSingleSlopeFunction = True Then
        ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
     End If
     '---------------------------------- db ----------------------
     Form3.Data1.Recordset("KeyNo") = KeyNum
     Form3.Data1.Recordset("Xcoord") = XValue
     Form3.Data1.Recordset("Ycoord") = YValue
     Form3.Data1.Recordset("Zcoord") = ZValue
     Form3.Data1.Recordset("Direction_Vector") = 16
     Form3.Data1.Recordset("QuadNo") = QuadNum
     Form3.Data1.Recordset("Frame#") = Fnumb
     Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
     ST1$ = Right$(ASCFileName$, 12)
     ST2$ = Left$(ST1$, 8) + ".VUE"
     Form3.Data1.Recordset("VUE_Filename") = ST2$
     Mpgfilename$ = GetMPGFilename()
     Form3.Data1.Recordset("Stream_File") = Mpgfilename$
     KeyNum = KeyNum + 1
     Form3.Data1.Recordset.Update
     '------------------------------ end db -----------------------------
     XValue = XValue - KeyScale
     Fnumb = FrameGroup + Fnumb
     YValue = YValue + KeyScale
     ZF1Value = ZF1Value + 1
  Loop
  ZF1Value = ZF1Value + 1
Next N
NYs = ((DX - KeyScale) / KeyScale) + 1
YValue = YLast + KeyScale
XValue = X4
ZF1Value = ZEValue
For N = 1 To NYs
  ASCFileName$ = GetNextFileString()
  Fnumb = 0
  Do Until YValue > Y3
  '-------------------------------------------
    Form3.Data1.Recordset.AddNew
  '-------------------------------------------
     If ZSingleSlopeFunction = True Then
```

```
            ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
          End If
          '---------------------------------- db -----------------------
          Form3.Data1.Recordset("KeyNo") = KeyNum
          Form3.Data1.Recordset("Xcoord") = XValue
          Form3.Data1.Recordset("Ycoord") = YValue
          Form3.Data1.Recordset("Zcoord") = ZValue
          Form3.Data1.Recordset("Direction_Vector") = 16
          Form3.Data1.Recordset("QuadNo") = QuadNum
          Form3.Data1.Recordset("Frame#") = Fnumb
          Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
          ST1$ = Right$(ASCFileName$, 12)
          ST2$ = Left$(ST1$, 8) + ".VUE"
          Form3.Data1.Recordset("VUE_Filename") = ST2$
          Mpgfilename$ = GetMPGFilename()
          Form3.Data1.Recordset("Stream_File") = Mpgfilename$
          KeyNum = KeyNum + 1
          Form3.Data1.Recordset.Update
          '------------------------------ end db ----------------------------
          XValue = XValue - KeyScale
          Fnumb = FrameGroup + Fnumb
          YValue = YValue + KeyScale
          ZF1Value = ZF1Value + 1
        Loop
        YValue = YLast + (N * KeyScale) + KeyScale
        ZF1Value = ZEValue + N
        XValue = X4
        Next N
        '------------------ db -------------------
        Form3.Data1.Recordset.Close
        '-------------------- db end ------------
End Sub

Sub SkewSWMulti () '// Builds SW motion vectors.
    RootFN = "SKSW"
    DY = Y2 - Y1
    DX = X3 - X2
    Fnumb = 0
    If ZStepFunction = True Then
       ZValue = ZAValue + ZBValue
    End If
    If ZConstantFunction = True Then
       ZValue = Z1
    End If
    '------------------ db -----------------
    Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
    Form3.Data1.RecordSource = "Quadrant"
    Form3.Data1.Refresh
    If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
    End If
    '------------------ end db --------------------
```

```
    For N = 1 To (DX / KeyScale)
      ASCFileName$ = GetNextFileString()
      Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
      Open ASCFileName$ For Append As #1
      Print #1, Header$
      Debug.Print Header$
      Fnumb = 0
      XValue = X1 + (N * KeyScale)
      YValue = Y2
      ZF1Value = ZFValue
      Do Until (XValue < X1)
      '-------------------------------------------
        Form3.Data1.Recordset.AddNew
      '-------------------------------------------
        If ZSingleSlopeFunction = True Then
          ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
       'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '---------------------------------- db ----------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 17
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db -----------------------------
        XValue = XValue - KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
      Loop
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
```

```
    Close #1
  Next N
  ZF1Value = ZFValue
  NS = (DY - DX) / KeyScale
  For N = 1 To NS
    XValue = X3
    YValue = Y3 - (N * KeyScale)
    YLast = YValue
    ASCFileName$ = GetNextFileString()
    Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
    Open ASCFileName$ For Append As #1
    Print #1, Header$
    'Debug.Print Header$
    Fnumb = 0
    Do Until XValue < X1
    '-------------------------------------------
      Form3.Data1.Recordset.AddNew
    '-------------------------------------------
      If ZSingleSlopeFunction = True Then
        ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
      End If
      OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
      OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
      Print #1, OutStrNext$
      'Debug.Print OutStrNext$
      Print #1, OutStrCBTF$
      '---------------------------------- db ----------------------
      Form3.Data1.Recordset("KeyNo") = KeyNum
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 17
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1
      Form3.Data1.Recordset.Update
      ''------------------------------ end db -----------------------------
      XValue = XValue - KeyScale
      Fnumb = FrameGroup + Fnumb
      YValue = YValue - KeyScale
      ZF1Value = ZF1Value - 1
  Loop
  OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
  OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
```

```
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      Debug.Print "End"
      Close #1
      ZF1Value = ZFValue
    Next N
    XValue = X4
   For N = 1 To (DX / KeyScale) - 1
   YValue = YLast - (N * KeyScale)
   ZF1Value = ZFValue - 1
     ASCFileName$ = GetNextFileString()
     Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
     Open ASCFileName$ For Append As #1
     Print #1, Header$
     'Debug.Print Header$
     Fnumb = 0
     Do Until YValue < Y1
     '-----------------------------------------
       Form3.Data1.Recordset.AddNew
     '-----------------------------------------
        If ZSingleSlopeFunction = True Then
           ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
        OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " +
Str$(Fnumb)
        Print #1, OutStrNext$
        'Debug.Print OutStrNext$
        Print #1, OutStrCBTF$
        '---------------------------------- db ------------------------
         Form3.Data1.Recordset("KeyNo") = KeyNum
         Form3.Data1.Recordset("Xcoord") = XValue
         Form3.Data1.Recordset("Ycoord") = YValue
         Form3.Data1.Recordset("Zcoord") = ZValue
         Form3.Data1.Recordset("Direction_Vector") = 17
         Form3.Data1.Recordset("QuadNo") = QuadNum
         Form3.Data1.Recordset("Frame#") = Fnumb
         Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
         ST1$ = Right$(ASCFileName$, 12)
         ST2$ = Left$(ST1$, 8) + ".VUE"
         Form3.Data1.Recordset("VUE_Filename") = ST2$
         Mpgfilename$ = GetMPGFilename()
         Form3.Data1.Recordset("Stream_File") = Mpgfilename$
         KeyNum = KeyNum + 1
         Form3.Data1.Recordset.Update
        '------------------------------ end db ----------------------------
        XValue = XValue - KeyScale
        Fnumb = FrameGroup + Fnumb
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value - 1
```

```
      Loop
      OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
      OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
      Print #1, OutStrScale$
      Print #1, OutStrRotate$
      Print #1, "End"
      'Debug.Print "End"
      Close #1
      ZF1Value = ZFValue - (N + 1)
      XValue = X4
      Next N
      '---------- db ----------
      Form3.Data1.Recordset.Close
      '---------- db end ----------
End Sub

Sub SkewSWMultiB () ' // ---------- SouthWest Skew motion vectors in backwards position
   RootFN = "SWBK"
   DY = Y2 - Y1
   DX = X3 - X2
   Fnumb = 0
   If ZStepFunction = True Then
      ZValue = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '---------- db ----------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '---------- end db ----------
   For N = 1 To (DX / KeyScale)
      ASCFileName$ = GetNextFileString()
      Fnumb = 0
      XValue = X1
      YValue = Y2 - (N * KeyScale)
      YLast = YValue
      ZF1Value = ZFValue - N
      Do Until (XValue > X1 + (N * KeyScale))
      '-----------------------------------------
        Form3.Data1.Recordset.AddNew
      '-----------------------------------------
        If ZSingleSlopeFunction = True Then
           ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
        End If
        '---------- db ----------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
```

```
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 18
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1
      Form3.Data1.Recordset.Update
      '------------------------------ end db -----------------------------
      XValue = XValue + KeyScale
      Fnumb = FrameGroup + Fnumb
      YValue = YValue + KeyScale
      ZF1Value = ZF1Value + 1
  Loop
Next N
'--------------------- 1st 3rd -------------------
ZF1Value = ZEValue - 1
NS = (DY - DX) / KeyScale
For N = 1 To NS
YValue = YLast - (N * KeyScale)
XValue = X1
ASCFileName$ = GetNextFileString()
Fnumb = 0
  Do Until XValue > X3
  '-------------------------------------------
    Form3.Data1.Recordset.AddNew
  '-------------------------------------------
    If ZSingleSlopeFunction = True Then
       ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
    End If
    '------------------------- db ---------------------------
    Form3.Data1.Recordset("KeyNo") = KeyNum
    Form3.Data1.Recordset("Xcoord") = XValue
    Form3.Data1.Recordset("Ycoord") = YValue
    Form3.Data1.Recordset("Zcoord") = ZValue
    Form3.Data1.Recordset("Direction_Vector") = 18
    Form3.Data1.Recordset("QuadNo") = QuadNum
    Form3.Data1.Recordset("Frame#") = Fnumb
    Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
    ST1$ = Right$(ASCFileName$, 12)
    ST2$ = Left$(ST1$, 8) + ".VUE"
    Form3.Data1.Recordset("VUE_Filename") = ST2$
    Mpgfilename$ = GetMPGFilename()
    Form3.Data1.Recordset("Stream_File") = Mpgfilename$
    KeyNum = KeyNum + 1
    Form3.Data1.Recordset.Update
    '------------------------------ end db ----------------------
```

```
            XValue = XValue + KeyScale
            Fnumb = FrameGroup + Fnumb
            YValue = YValue + KeyScale
            ZF1Value = ZF1Value + 1
          Loop
          ZF1Value = ZEValue - 1
        Next N
        '------------------ Next 3rd ------------------
      YValue = Y1
      YEndValue = Y3 - KeyScale
      For N = 1 To (DX / KeyScale) - 1
        XValue = X1 + (N * KeyScale)
        ASCFileName$ = GetNextFileString()
        Fnumb = 0
        Do Until XValue > X4
        '-----------------------------------------
          Form3.Data1.Recordset.AddNew
        '-----------------------------------------
          If ZSingleSlopeFunction = True Then
             ZValue = Z1 + ((ZAValue / ZBValue) * (ZF1Value * KeyScale))
          End If
          '------------------------- db --------------------------
          Form3.Data1.Recordset("KeyNo") = KeyNum
          Form3.Data1.Recordset("Xcoord") = XValue
          Form3.Data1.Recordset("Ycoord") = YValue
          Form3.Data1.Recordset("Zcoord") = ZValue
          Form3.Data1.Recordset("Direction_Vector") = 18
          Form3.Data1.Recordset("QuadNo") = QuadNum
          Form3.Data1.Recordset("Frame#") = Fnumb
          Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
          ST1$ = Right$(ASCFileName$, 12)
          ST2$ = Left$(ST1$, 8) + ".VUE"
          Form3.Data1.Recordset("VUE_Filename") = ST2$
          Mpgfilename$ = GetMPGFilename()
          Form3.Data1.Recordset("Stream_File") = Mpgfilename$
          KeyNum = KeyNum + 1
          Form3.Data1.Recordset.Update
          '------------------------------ end db ----------------------
          XValue = XValue + KeyScale
          Fnumb = FrameGroup + Fnumb
          YValue = YValue + KeyScale
          ZF1Value = ZF1Value + 1
        Loop
        YValue = Y1
        ZF1Value = ZEValue - 1
        XValue = XValue + KeyScale
        Next N
        '----------------- db -------------------
        Form3.Data1.Recordset.Close
        '------------------- db end ------------
End Sub
```

```
Sub XForwardProc () '/// Builds X forward motion vectors
   XValue = X2
   YValue = Y2
   DY1 = Y3 - Y4
   ZF1Value = (DY1 - (Y3 - Y2)) / KeyScale
   Fnumb = 0
   '-------------------------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '-------------------------------------
   If ZStepFunction = True Then
      Z1 = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '-------------------------------------
   ASCFileName$ = GetNextFileString()
   Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
   Open ASCFileName$ For Append As #1
   Print #1, Header$
   Do Until XValue > X3
   '-------------------------------------
   Form3.Data1.Recordset.AddNew
   '-------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + (ZAValue / ZBValue * (ZF1Value * KeyScale))
      End If
      OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
      OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " + Str$(Fnumb)
      '-------------------------------
      Form3.Data1.Recordset("KeyNo") = KeyNum
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 1
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
      Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
      ST1$ = Right$(ASCFileName$, 12)
      ST2$ = Left$(ST1$, 8) + ".VUE"
      Form3.Data1.Recordset("VUE_Filename") = ST2$
      Mpgfilename$ = GetMPGFilename()
      Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      KeyNum = KeyNum + 1'----- db
      XValue = XValue + KeyScale
      Fnumb = FrameGroup + Fnumb
```

```
      Print #1, OutStrNext$
      Print #1, OutStrCBTF$
      Form3.Data1.Recordset.Update
   Loop
   OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
   OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
   Print #1, OutStrScale$
   Print #1, OutStrRotate$
   Print #1, "End"
   Close #1
   Form3.Data1.Recordset.Close
End Sub

Sub XForwardProcB () ' // ----- build X forward motion vectors in backward direction.

   XValue = X2
   YValue = Y2
   DY1 = Y3 - Y4
   ZF1Value = (DY1 - (Y3 - Y2)) / KeyScale
   Fnumb = 0
   '-------------------------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '-------------------------------------
   If ZStepFunction = True Then
      Z1 = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '-------------------------------------
   ASCFileName$ = GetNextFileString()
   Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
   Do Until XValue > X3
   '-------------------------------------
   Form3.Data1.Recordset.AddNew
   '-------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + (ZAValue / ZBValue * (ZF1Value * KeyScale))
      End If
      '-------------------------------------
      Form3.Data1.Recordset("KeyNo") = KeyNum
      Form3.Data1.Recordset("Xcoord") = XValue
      Form3.Data1.Recordset("Ycoord") = YValue
      Form3.Data1.Recordset("Zcoord") = ZValue
      Form3.Data1.Recordset("Direction_Vector") = 4
      Form3.Data1.Recordset("QuadNo") = QuadNum
      Form3.Data1.Recordset("Frame#") = Fnumb
```

```
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1'----- db
        XValue = XValue + KeyScale
        Fnumb = FrameGroup + Fnumb
        Form3.Data1.Recordset.Update
    Loop
    Form3.Data1.Recordset.Close
End Sub

Sub XForwardProcMulti ()
      DY = Y2 - Y1 + KeyScale
      Do While DY > 0
         Call XForwardProc
         DY = DY - KeyScale
         Y2 = Y2 - KeyScale
      Loop
End Sub

Sub XForwardProcMultiB ()
      DY = Y2 - Y1 + KeyScale
      Do While DY > 0
         Call XReturnProcB
         DY = DY - KeyScale
         Y2 = Y2 - KeyScale
      Loop
End Sub

Sub XReturnProc ()
   XValue = X3
   YValue = Y2
   DY = Y3 - Y4
   ZF1Value = (DY - (Y3 - Y2)) / KeyScale
   Fnumb = 0
   If ZStepFunction = True Then
      Z1 = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '-------------------------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '-------------------------------------
```

```
    ASCFileName$ = GetNextFileString()
    Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
    Open ASCFileName$ For Append As #1
    Print #1, Header$
     Do Until XValue < X2
       '---------------------------------------
       Form3.Data1.Recordset.AddNew
       '---------------------------------------
       If ZSingleSlopeFunction = True Then
          ZValue = Z1 + (ZAValue / ZBValue * (ZF1Value * KeyScale))
       End If
       OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " +                                                    Format$(ZValue,
"###.00") + ", " + Str$(Fnumb)
       OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " + Str$(Fnumb)
      '-----------------------------------------
       Form3.Data1.Recordset("KeyNo") = KeyNum
       Form3.Data1.Recordset("Xcoord") = XValue
       Form3.Data1.Recordset("Ycoord") = YValue
       Form3.Data1.Recordset("Zcoord") = ZValue
       Form3.Data1.Recordset("Direction_Vector") = 3
       Form3.Data1.Recordset("QuadNo") = QuadNum
       Form3.Data1.Recordset("Frame#") = Fnumb
       Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
       ST1$ = Right$(ASCFileName$, 12)
       ST2$ = Left$(ST1$, 8) + ".VUE"
       Form3.Data1.Recordset("VUE_Filename") = ST2$
       Mpgfilename$ = GetMPGFilename()
       Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      '-------------------------------------------------
       XValue = XValue - KeyScale
       KeyNum = KeyNum + 1'----- db
       Fnumb = FrameGroup + Fnumb
       Print #1, OutStrNext$
       Print #1, OutStrCBTF$
       Form3.Data1.Recordset.Update
     Loop
    OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
    OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
    Print #1, OutStrScale$
    Print #1, OutStrRotate$
    Print #1, "End"
    Close #1
    Form3.Data1.Recordset.Close
End Sub

Sub XReturnProcB ()
    XValue = X3
    YValue = Y2
    DY = Y3 - Y4
    ZF1Value = (DY - (Y3 - Y2)) / KeyScale
    Fnumb = 0
```

```
    If ZStepFunction = True Then
       Z1 = ZAValue + ZBValue
    End If
    If ZConstantFunction = True Then
       ZValue = Z1
    End If
    '---------------------------------------
    Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
    Form3.Data1.RecordSource = "Quadrant"
    Form3.Data1.Refresh
    If Form3.Data1.Recordset.BOF = False Then
       Form3.Data1.Recordset.MoveLast
    End If
    '---------------------------------------
    ASCFileName$ = GetNextFileString()
     Do Until XValue < X2
       '---------------------------------------
       Form3.Data1.Recordset.AddNew
       '---------------------------------------
       If ZSingleSlopeFunction = True Then
          ZValue = Z1 + (ZAValue / ZBValue * (ZF1Value * KeyScale))
       End If
      '-----------------------------------------------
       Form3.Data1.Recordset("KeyNo") = KeyNum
       Form3.Data1.Recordset("Xcoord") = XValue
       Form3.Data1.Recordset("Ycoord") = YValue
       Form3.Data1.Recordset("Zcoord") = ZValue
       Form3.Data1.Recordset("Direction_Vector") = 2
       Form3.Data1.Recordset("QuadNo") = QuadNum
       Form3.Data1.Recordset("Frame#") = Fnumb
       Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
       ST1$ = Right$(ASCFileName$, 12)
       ST2$ = Left$(ST1$, 8) + ".VUE"
       Form3.Data1.Recordset("VUE_Filename") = ST2$
       Mpgfilename$ = GetMPGFilename()
       Form3.Data1.Recordset("Stream_File") = Mpgfilename$
      '-----------------------------------------------------
       XValue = XValue - KeyScale
       KeyNum = KeyNum + 1'----- db
       Fnumb = FrameGroup + Fnumb
       Form3.Data1.Recordset.Update
     Loop
    Form3.Data1.Recordset.Close
End Sub

Sub XReturnProcMulti () '// Build X in return motion vectors
    RootFN = "XRTN"
    DY = Y2 - Y1 + KeyScale
    Do While DY > 0
       Call XReturnProc
       DY = DY - KeyScale
       Y2 = Y2 - KeyScale
```

```
    Loop
End Sub

Sub XReturnProcMultiB () // Build X in return motion vectors Backwards
      DY = Y2 - Y1 + KeyScale
      Do While DY > 0
         Call XForwardProcB
         DY = DY - KeyScale
         Y2 = Y2 - KeyScale
      Loop
End Sub

Sub YForwardProc () ' Y forward Motion Vectors
   YValue = Y1
   XValue = X3
   Fnumb = 0
   If ZStepFunction = True Then
      Z1 = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '-----------------------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '-----------------------------------
   ASCFileName$ = GetNextFileString()
   Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
   Open ASCFileName$ For Append As #1
   Print #1, Header$
    Do Until YValue > Y2
      '---------------------------------------
      Form3.Data1.Recordset.AddNew
      '---------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + (ZAValue / ZBValue * (ZEValue * KeyScale))
      End If
      OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
      OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " + Str$(Fnumb)
         '---------------------------------- db ----------------------
         Form3.Data1.Recordset("KeyNo") = KeyNum
         Form3.Data1.Recordset("Xcoord") = XValue
         Form3.Data1.Recordset("Ycoord") = YValue
         Form3.Data1.Recordset("Zcoord") = ZValue
         Form3.Data1.Recordset("Direction_Vector") = 5
         Form3.Data1.Recordset("QuadNo") = QuadNum
         Form3.Data1.Recordset("Frame#") = Fnumb
```

```
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db ------------------------------
      YValue = YValue + KeyScale
      ZEValue = ZEValue + 1
      Fnumb = FrameGroup + Fnumb
      Print #1, OutStrNext$
      Print #1, OutStrCBTF$
    Loop
    OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
    OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
   Print #1, OutStrScale$
   Print #1, OutStrRotate$
   Print #1, "End"
   Close #1
      '----------------- db -------------------
      Form3.Data1.Recordset.Close
      '------------------- db end -------------
End Sub

Sub YForwardProcB () '// --- Y forward motion vectors.
   YValue = Y1
   XValue = X3
   Fnumb = 0
   If ZStepFunction = True Then
      Z1 = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '-------------------------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '-------------------------------------
   ASCFileName$ = GetNextFileString()
    Do Until YValue > Y2
      '---------------------------------------
      Form3.Data1.Recordset.AddNew
      '---------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + (ZAValue / ZBValue * (ZEValue * KeyScale))
      End If
```

```
            '---------------------------------- db ----------------------
            Form3.Data1.Recordset("KeyNo") = KeyNum
            Form3.Data1.Recordset("Xcoord") = XValue
            Form3.Data1.Recordset("Ycoord") = YValue
            Form3.Data1.Recordset("Zcoord") = ZValue
            Form3.Data1.Recordset("Direction_Vector") = 8
            Form3.Data1.Recordset("QuadNo") = QuadNum
            Form3.Data1.Recordset("Frame#") = Fnumb
            Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
            ST1$ = Right$(ASCFileName$, 12)
            ST2$ = Left$(ST1$, 8) + ".VUE"
            Form3.Data1.Recordset("VUE_Filename") = ST2$
            Mpgfilename$ = GetMPGFilename()
            Form3.Data1.Recordset("Stream_File") = Mpgfilename$
            KeyNum = KeyNum + 1
            Form3.Data1.Recordset.Update
            '------------------------------ end db ----------------------------
        YValue = YValue + KeyScale
        ZEValue = ZEValue + 1
        Fnumb = FrameGroup + Fnumb
      Loop
        '----------------- db -------------------
        Form3.Data1.Recordset.Close
        '------------------- db end -------------
End Sub

Sub YForwardProcMulti () ' Motion vectors for Y forward
    DX = X3 - X2
    Do Until DX < 0
        RootFN = "YFWD"
        Call YForwardProc
        DX = DX - KeyScale
        X3 = X3 - KeyScale
    Loop
End Sub

Sub YForwardProcMultiB ()
    DX = X3 - X2
    Do Until DX < 0
        RootFN = "YBWD"
        Call YReturnProcB
        DX = DX - KeyScale
        X3 = X3 - KeyScale
    Loop
End Sub

Sub YReturnProc ()
    YValue = Y2
    XValue = X3
    Fnumb = 0
    ZF1Value = ZEValue
    If ZStepFunction = True Then
```

```
      Z1 = ZAValue + ZBValue
    End If
    If ZConstantFunction = True Then
      ZValue = Z1
    End If
    '-------------------------------------
    Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
    Form3.Data1.RecordSource = "Quadrant"
    Form3.Data1.Refresh
    If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
    End If
    '-------------------------------------
    ASCFileName$ = GetNextFileString()
    Header$ = "Continue2, Auto-Generated Node Data, " + ASCFileName$
    Open ASCFileName$ For Append As #1
    Print #1, Header$
    Do Until YValue < Y1
      '-------------------------------------
      Form3.Data1.Recordset.AddNew
      '-------------------------------------
      If ZSingleSlopeFunction = True Then
        ZValue = Z1 + (ZAValue / ZBValue * ((ZFValue - ZF1Value) * KeyScale))
      End If
      OutStrNext$ = "Move, " + ObjectName + ", " + Str$(XValue) + ", " +
Str$(YValue) + ", " + Format$(ZValue, "###.00") + ", " + Str$(Fnumb)
      OutStrCBTF$ = "CBTFT, " + ObjectName + ", 0, 0, 25, 25, 0, 0, " + Str$(Fnumb)
        '---------------------------------- db ----------------------
        Form3.Data1.Recordset("KeyNo") = KeyNum
        Form3.Data1.Recordset("Xcoord") = XValue
        Form3.Data1.Recordset("Ycoord") = YValue
        Form3.Data1.Recordset("Zcoord") = ZValue
        Form3.Data1.Recordset("Direction_Vector") = 7
        Form3.Data1.Recordset("QuadNo") = QuadNum
        Form3.Data1.Recordset("Frame#") = Fnumb
        Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
        ST1$ = Right$(ASCFileName$, 12)
        ST2$ = Left$(ST1$, 8) + ".VUE"
        Form3.Data1.Recordset("VUE_Filename") = ST2$
        Mpgfilename$ = GetMPGFilename()
        Form3.Data1.Recordset("Stream_File") = Mpgfilename$
        KeyNum = KeyNum + 1
        Form3.Data1.Recordset.Update
        '------------------------------ end db -----------------------------
      YValue = YValue - KeyScale
      ZF1Value = ZF1Value + 1
      Fnumb = FrameGroup + Fnumb
      Print #1, OutStrNext$
      Print #1, OutStrCBTF$
    Loop
    OutStrScale$ = "Scale, " + ObjectName + ", " + "1, 1, 1, 0"
    OutStrRotate$ = "Rotate, " + ObjectName + ", " + "1, 0, 0, 0, 0"
```

```
    Print #1, OutStrScale$
    Print #1, OutStrRotate$
    Print #1, "End"
    Close #1
       '---------- db ----------
       Form3.Data1.Recordset.Close
       '---------- db end ----------
End Sub

Sub YReturnProcB ()
   YValue = Y2
   XValue = X3
   Fnumb = 0
   ZF1Value = ZEValue
   If ZStepFunction = True Then
      Z1 = ZAValue + ZBValue
   End If
   If ZConstantFunction = True Then
      ZValue = Z1
   End If
   '-----------------------------------
   Form3.Data1.DatabaseName = "D:\vb\mpgvr.mdb"
   Form3.Data1.RecordSource = "Quadrant"
   Form3.Data1.Refresh
   If Form3.Data1.Recordset.BOF = False Then
      Form3.Data1.Recordset.MoveLast
   End If
   '-----------------------------------
   ASCFileName$ = GetNextFileString()
   Do Until YValue < Y1
      '---------------------------------------
      Form3.Data1.Recordset.AddNew
      '---------------------------------------
      If ZSingleSlopeFunction = True Then
         ZValue = Z1 + (ZAValue / ZBValue * ((ZFValue - ZF1Value) * KeyScale))
      End If
         '---------------------------------- db ----------------------
         Form3.Data1.Recordset("KeyNo") = KeyNum
         Form3.Data1.Recordset("Xcoord") = XValue
         Form3.Data1.Recordset("Ycoord") = YValue
         Form3.Data1.Recordset("Zcoord") = ZValue
         Form3.Data1.Recordset("Direction_Vector") = 6
         Form3.Data1.Recordset("QuadNo") = QuadNum
         Form3.Data1.Recordset("Frame#") = Fnumb
         Form3.Data1.Recordset("ASC_Filename") = Right$(ASCFileName$, 12)
         ST1$ = Right$(ASCFileName$, 12)
         ST2$ = Left$(ST1$, 8) + ".VUE"
         Form3.Data1.Recordset("VUE_Filename") = ST2$
         Mpgfilename$ = GetMPGFilename()
         Form3.Data1.Recordset("Stream_File") = Mpgfilename$
         KeyNum = KeyNum + 1
         Form3.Data1.Recordset.Update
```

```
        '------------------------------ end db ------------------------------
        YValue = YValue - KeyScale
        ZF1Value = ZF1Value + 1
        Fnumb = FrameGroup + Fnumb
      Loop
        '----------------- db -------------------
        Form3.Data1.Recordset.Close
        '------------------- db end -------------
End Sub

Sub YReturnProcMulti ()
   DX = X3 - X2
   Do Until DX < 0
      RootFN = "YRTN"
      Call YReturnProc
      DX = DX - KeyScale
      X3 = X3 - KeyScale
   Loop
End Sub

Sub YReturnProcMultiB ()
   DX = X3 - X2
   Do Until DX < 0
      RootFN = "YBRT"
      Call YForwardProcB
      DX = DX - KeyScale
      X3 = X3 - KeyScale
   Loop
End Sub
```

```
'// -- Player program main functions ----- ///
Global Dir_Vec As Integer
Global XValue As Integer
Global YValue As Integer
Global RecPointerNum As Long
Global Tblno As Integer
Type StateTable '// Current state table structure
   CPV As Integer
   MV  As Integer
   NPV As Integer
   NPCC As Integer
   TblName As String * 2
End Type
Type StateRegister
   CurXPos As Integer
   CurYPos As Integer
   CurZPos As Integer
   CurPV As Integer
   Bounds As Integer
   QuanNum As Integer
End Type
Global StateMB(8) As StateTable
Global StateMF(8) As StateTable
Global StateTR(8) As StateTable
Global StateTL(8) As StateTable
Global StateVar(8) As StateTable
Global CurrentState As StateRegister

Sub LoadStateTables () '// ----------> Load state tables from file on HD
   Dim Filenum, NumberSub As Integer
   Dim RecordLen, Position As Long
   RecordLen = Len(StateVar(1))
   Filenum = FreeFile
   Position = 1
   Open "D:/VB/KEYFRAME/STATETBL.FIL" For Random As Filenum Len = RecordLen
    For M = 1 To 4
        Get #Filenum, Position, StateVar(1)
        TableName$ = StateVar(1).TblName
        Select Case TableName$
            Case "MB"
             For N = 1 To 8
                Get #Filenum, Position, StateMB(N)
                Position = Position + 1
             Next N
            Case "MF"
             For N = 1 To 8
                Get #Filenum, Position, StateMF(N)
                Position = Position + 1
             Next N
            Case "TL"
             For N = 1 To 8
```

```
                Get #Filenum, Position, StateTL(N)
                Position = Position + 1
             Next N
            Case "TR"
             For N = 1 To 8
                Get #Filenum, Position, StateTR(N)
                Position = Position + 1
             Next N
        End Select
   Next M
   Close #Filenum
End Sub

Sub PrintStateTbl () ' // ---> Show state table information to  table editor
   Dim Filenum, NumberSub As Integer
   Dim RecordLen, Position As Long
   RecordLen = Len(StateVar(1))
   Filenum = FreeFile
   Position = Val((StateBuild.Text2.Text) - 1) * 8
   Open "D:/VB/KEYFRAME/STATETBL.FIL" For Random As Filenum Len = RecordLen
   For N = 1 To 8
        Position = Position + 1
        Get #Filenum, Position, StateVar(N)
        StateBuild.Grid1.Row = N
        StateBuild.Grid1.Col = 0
        StateBuild.Grid1.Text = Str$(StateVar(N).CPV)
        StateBuild.Grid1.Col = 1
        StateBuild.Grid1.Text = Str$(StateVar(N).MV)
        StateBuild.Grid1.Col = 2
        StateBuild.Grid1.Text = Str$(StateVar(N).NPV)
        StateBuild.Grid1.Col = 3
        StateBuild.Grid1.Text = Str$(StateVar(N).NPCC)
        NumberSub = N
   Next N
   StateBuild.Label3.Caption = "State Table = " + StateVar(NumberSub).TblName
   Close #Filenum
End Sub

Sub SaveTable () ' // Saves edited state table to file
   Dim LastRecord As Long
   Dim Filenum As Integer
   Dim RecordLen As Long
   RecordLen = Len(StateVar(1))
   Filenum = FreeFile
   LastRecord = (Val(StateBuild.Text2.Text) - 1) * 8
   Open "D:/VB/KEYFRAME/STATETBL.FIL" For Random As Filenum Len = RecordLen
   StateName = Form2.Text1.Text
   For N = 1 To 8
        LastRecord = LastRecord + 1
        StateBuild.Grid1.Row = N
        StateBuild.Grid1.Col = 0
```

```
        StateVar(N).CPV = Val(StateBuild.Grid1.Text)
        StateBuild.Grid1.Col = 1
        StateVar(N).MV = Val(StateBuild.Grid1.Text)
        StateBuild.Grid1.Col = 2
        StateVar(N).NPV = Val(StateBuild.Grid1.Text)
        StateBuild.Grid1.Col = 3
        StateVar(N).NPCC = Val(StateBuild.Grid1.Text)
        StateVar(N).TblName = StateName
        Put #Filenum, LastRecord, StateVar(N)
    Next N
    Close #Filenum
End Sub
```

```
'// --- Player Program Form functions --- > //
Declare Sub CWsensorThere Lib "d:\VB\VBWND\CWAND.DLL" (ByVal PORTNO%, HatCal
As Integer)
Declare Sub CWgetStatus Lib "d:\VB\VBWND\CWAND.DLL" (ByVal PORTNO%, HATVL As
Integer, BUTTONVL As Integer)
Global CalValOk As Integer
Global TestOk As Integer
Global Dir_Vec As Integer
Global XValue As Integer
Global YValue As Integer
Global RecPointerNum As Long
Global Tblno As Integer
Global EventState As Integer
Type StateTable
   CPV As Integer
   MV  As Integer
   NPV As Integer
   NPCC As Integer
   TblName As String * 2
End Type
Type StateRegister
   CurXPos As Integer
   CurYPos As Integer
   CurZPos As Integer
   CurPV As Integer
   Bounds As Integer
   QuanNum As Integer
End Type
Global StateMB(8) As StateTable
Global StateMF(8) As StateTable
Global StateTR(8) As StateTable
Global StateTL(8) As StateTable
Global StateVar(8) As StateTable
Global CurrentState As StateRegister
Global XP1 As Integer
Global XP3 As Integer
Global YP2 As Integer
Global YP1 As Integer
Global StateTableName$
Global WandButtonState$
Global ProjectorState As Integer
Global MPGFILE$

Sub EventHandler () '// ---> Simple event handler for eventual event editor
   If EventState = True Then
      Windname$ = Form1.hWnd
      mcicomp$ = "WINDOW DWELL HANDLE " & Windname$
      Form1.mkMci3.Send = mcicomp$
      Form1.mkMci3.Send = "PUT DWELL DESTINATION AT 75 14 704 485"

      Form1.mkMci3.Send = "PLAY DWELL REPEAT"
   End If
```

```
    If EventState = False Then
       Form1.mkMci3.Send = "STOP DWELL"
    End If
End Sub

Sub InitState () '// Initialize state to position 156,  24
    Form1.Text6.Text = "156"
    Form1.Text7.Text = "24"
    Form1.Text8.Text = "5"
    Form1.Text9.Text = "1"
    ProjectorState = 1
End Sub

Sub LoadStateTables () '// ---------> get operational state tables
    Dim Filenum, NumberSub As Integer
    Dim RecordLen, Position As Long
    RecordLen = Len(StateVar(1))
    Filenum = FreeFile
    Position = 1
    Open "D:/VB/NAVGATE/STATETBL.FIL" For Random As Filenum Len = RecordLen
     For M = 1 To 4
          Get #Filenum, Position, StateVar(1)
          TableName$ = StateVar(1).TblName
          Select Case TableName$
               Case "MB"
                For N = 1 To 8
                   Get #Filenum, Position, StateMB(N)
                   Position = Position + 1
                Next N
               Case "MF"
                For N = 1 To 8
                   Get #Filenum, Position, StateMF(N)
                   Position = Position + 1
                Next N
               Case "TL"
                For N = 1 To 8
                   Get #Filenum, Position, StateTL(N)
                   Position = Position + 1
                Next N
               Case "TR"
                For N = 1 To 8
                   Get #Filenum, Position, StateTR(N)
                   Position = Position + 1
                Next N
          End Select
    Next M
    Close #Filenum
End Sub

Sub PlayVector1 () ' // Virtual projector1 player routine
    FilePathName$ = "D:\VB\MPGS\" + Form1.Text1.Text
    SendString$ = "OPEN " + FilePathName$ + " ALIAS MVEC STYLE POPUP"
```

```
    Form1.mkMci1.Send = SendString$
    Windname$ = Form1.hWnd
    mcicomp$ = "WINDOW MVEC HANDLE " & Windname$
    Form1.mkMci1.Send = mcicomp$
    Form1.mkMci1.Send = "PUT MVEC DESTINATION AT 75 14 704 485"
   Form1.mkMci1.Send = "PLAY MVEC WAIT"
    ProjectorState = 2
    Windname1$ = Form3.hWnd
    mcicomp1$ = "WINDOW MVEC1 HANDLE " & Windname1$
   Form1.mkMci1.Send = mcicomp1$
   Form1.mkMci1.Send = "PUT MVEC1 DESTINATION AT 75 14 704 485"
   Form1.mkMci1.Send = "CLOSE MVEC1"
End Sub

Sub PlayVector2 () '// Virtual projector2 player routine
    FilePathName$ = "D:\VB\MPGS\" + Form1.Text1.Text
    SendString$ = "OPEN " + FilePathName$ + " ALIAS MVEC1 STYLE POPUP"
   Form1.mkMci2.Send = SendString$
    Windname$ = Form1.hWnd
    mcicomp$ = "WINDOW MVEC1 HANDLE " & Windname$
   Form1.mkMci2.Send = mcicomp$
   Form1.mkMci2.Send = "PUT MVEC1 DESTINATION AT 75 14 704 485"
   Form1.mkMci2.Send = "PLAY MVEC1 WAIT"
    ProjectorState = 1
    Windname1$ = Form3.hWnd
    mcicomp1$ = "WINDOW MVEC HANDLE " & Windname1$
   Form1.mkMci1.Send = mcicomp1$
   Form1.mkMci1.Send = "PUT MVEC DESTINATION AT 75 14 704 485"
   Form1.mkMci1.Send = "CLOSE MVEC"
End Sub

Sub PlayWand () '// Execute poll routine for Wand Controller.
    PORTNO1% = 1
    TestOk = False
    HatValue% = 999
    ButtonValue% = 999
    HatCal% = 0
      Call CWgetStatus(PORTNO1%, HatValue%, ButtonValue%)
      Select Case ButtonValue%
         Case 1
            WandButtonState = "Trigger"
         Case 2
            WandButtonState = "Top"
         Case 4
            WandButtonState = "Thumb"
         Case 8
            WandButtonState = "Pinky"
         Case Else
            WandButtonState = "Button"
      End Select
      Select Case HatValue%
         Case 1
```

```
            StateTableName$ = "MF"
          Case 2
            StateTableName$ = "MB"
          Case 3
            StateTableName$ = "TL"
          Case 4
             StateTableName$ = "TR"
          Case Else
             StateTableName$ = "CT"
       End Select
'-------------------------------------------------------------------------------------
End Sub

Sub TestWand () '// Diagnostic routine for wand controller.
    PORTNO1% = 1
    TestOk = False
    HatValue% = 999
    ButtonValue% = 999
    HatCal% = 0
      Do Until TestOk = True
      Call CWgetStatus(PORTNO1%, HatValue%, ButtonValue%)
      Select Case ButtonValue%
         Case 1
           Calibrate.ButtonPan.Caption = "Trigger"
         Case 2
           Calibrate.ButtonPan.Caption = "Top"
         Case 4
           Calibrate.ButtonPan.Caption = "Thumb"
         Case 8
           Calibrate.ButtonPan.Caption = "Pinky"
         Case Else
            Calibrate.ButtonPan.Caption = "BUTTON"
      End Select
      Select Case HatValue%
        Case 1
          Calibrate.Picture1.Picture = LoadPicture("d:\vb\vbwnd\fore.bmp")
        Case 2
          Calibrate.Picture1.Picture = LoadPicture("d:\vb\vbwnd\back.bmp")
        Case 3
          Calibrate.Picture1.Picture = LoadPicture("d:\vb\vbwnd\left.bmp")
        Case 4
          Calibrate.Picture1.Picture = LoadPicture("d:\vb\vbwnd\right.bmp")
        Case Else
          Calibrate.Picture1.Picture = LoadPicture("d:\vb\vbwnd\center.bmp")
      End Select
        DoEvents
    Loop
End Sub

Sub UpdateStates () '// Update state table in memory based on new wand data
  K = Val(Form1.Text2.Text)
  CPValue = Val(Form1.Text8.Text)
```

```
XP1 = Val(Form1.Text3.Text)
XP3 = Val(Form1.Text10.Text)
YP1 = Val(Form1.Text4.Text)
YP3 = Val(Form1.Text11.Text)
Select Case StateTableName$
   Case "MB"
   For N = 1 To 8
   If CPValue = StateMB(N).CPV Then
      MoveVector = StateMB(N).MV
      NextPV = StateMB(N).NPV
      NextPOSC = StateMB(N).NPCC
   End If
   Next N
   Case "MF"
   For N = 1 To 8
   If CPValue = StateMF(N).CPV Then
      MoveVector = StateMF(N).MV
      NextPV = StateMF(N).NPV
      NextPOSC = StateMF(N).NPCC
   End If
   Next N
   Case "TL"
   For N = 1 To 8
   If CPValue = StateTL(N).CPV Then
      MoveVector = StateTL(N).MV
      NextPV = StateTL(N).NPV
      NextPOSC = StateTL(N).NPCC
   End If
   Next N
   Case "TR"
   For N = 1 To 8
   If CPValue = StateTR(N).CPV Then
      MoveVector = StateTR(N).MV
      NextPV = StateTR(N).NPV
      NextPOSC = StateTR(N).NPCC
   End If
   Next N
   End Select
   XNext = Val(Form1.Text6.Text)
   YNext = Val(Form1.Text7.Text)
   Select Case NextPOSC
         Case 0
         Case 1
            XNext = Val(Form1.Text6.Text) + K
         Case 2
            XNext = Val(Form1.Text6.Text) - K
         Case 3
            XNext = Val(Form1.Text6.Text) + K
            YNext = Val(Form1.Text7.Text) + K
         Case 4
            XNext = Val(Form1.Text6.Text) - K
            YNext = Val(Form1.Text7.Text) - K
```

```
            Case 5
               YNext = Val(Form1.Text7.Text) + K
            Case 6
               YNext = Val(Form1.Text7.Text) - K
            Case 7
               XNext = Val(Form1.Text6.Text) + K
               YNext = Val(Form1.Text7.Text) - K
            Case 8
               XNext = Val(Form1.Text6.Text) - K
               YNext = Val(Form1.Text7.Text) + K
      End Select
      If XNext > XP3 Or XNext < XP1 Then
         Beep
         MPGFILE$ = "NO File"
         Exit Sub
      End If
      If YNext > YP3 Or YNext < YP1 Then
         Beep
         MPGFILE$ = "NO File"
         Exit Sub
      End If
      Form1.Text8.Text = Str$(NextPV)
      Dir_Vector$ = Str$(MoveVector)
      XC$ = Form1.Text6.Text
      YC$ = Form1.Text7.Text
      SQL1$ = "Select Stream_File FROM QUADRANT where Direction_Vector =  " +
Dir_Vector$
      SQL2$ = " AND Xcoord = " + XC$ + "AND Ycoord = " + YC$
      SQL3$ = SQL1$ + SQL2$
      Form1.Data1.RecordSource = SQL3$
      Form1.Data1.Refresh
      FX$ = Form1.Text1.Text
      FN$ = "D:\VB\MPGS\" + FX$
      On Error GoTo ERRORHandler
      R = FileLen(FN$)
      Form1.Text6.Text = Str$(XNext)
      Form1.Text7.Text = Str$(YNext)
      MPGFILE$ = ""
GetOUT: Exit Sub
ERRORHandler:
      MPGFILE$ = "NO File"
      Resume GetOUT
End Sub
```

CLAIMS

What is claimed is:

1. A method for generating a navigable virtual space, comprising the steps of:

generating a three dimensional model of the space;

imposing a coordinate system on the model, the coordinate system including a plurality of discrete points;

generating one or more motion vectors associated with each point, each motion vector representing a translation to a different point, or a rotation about a single point; and

rendering and encoding a video sequence associated with each motion vector, each video sequence showing the translation or rotation represented by each motion vector.

2. In a computer system including a database of motion vectors and associated encoded video sequences, each motion vector representing a particular rotation or translation within a virtual space, each video sequence showing the translation or rotation represented by its associated motion vector, and each video sequence having a last frame identical to a first frame of a subsequent video sequence,

a method for displaying a concatenated series of video sequences, comprising the steps of:

playing a first video sequence via a first virtual projector;

queuing a subsequent video sequence with a second virtual projector, the second virtual projector synchronized with the first virtual projector;

beginning to play the first frame of the subsequent video sequence via the second virtual projector when the last frame of the first video sequence is played by the first virtual projector.

3. In a computer system including a database of motion vectors and associated encoded video sequences, each motion vector representing a particular rotation or translation within a virtual space, and each video sequence showing the translation or rotation represented by its associated motion vector,

a method for generating a navigational framework of the virtual space, comprising the steps of:

defining the number of degrees of freedom allowable for an operator navigating through the virtual space, the operator having a position, orientation, and velocity in the virtual space;

defining areas within the virtual space in which an operator is permitted to navigate; and

generating a state machine for determining which particular motion vector should be accessed from the database at any given time, the determination of the state machine based upon the number of degrees of freedom allowable

the operator navigating through the virtual space, the position, orientation, and velocity of the operator in the virtual space, and the definition of navigable areas within the virtual space.

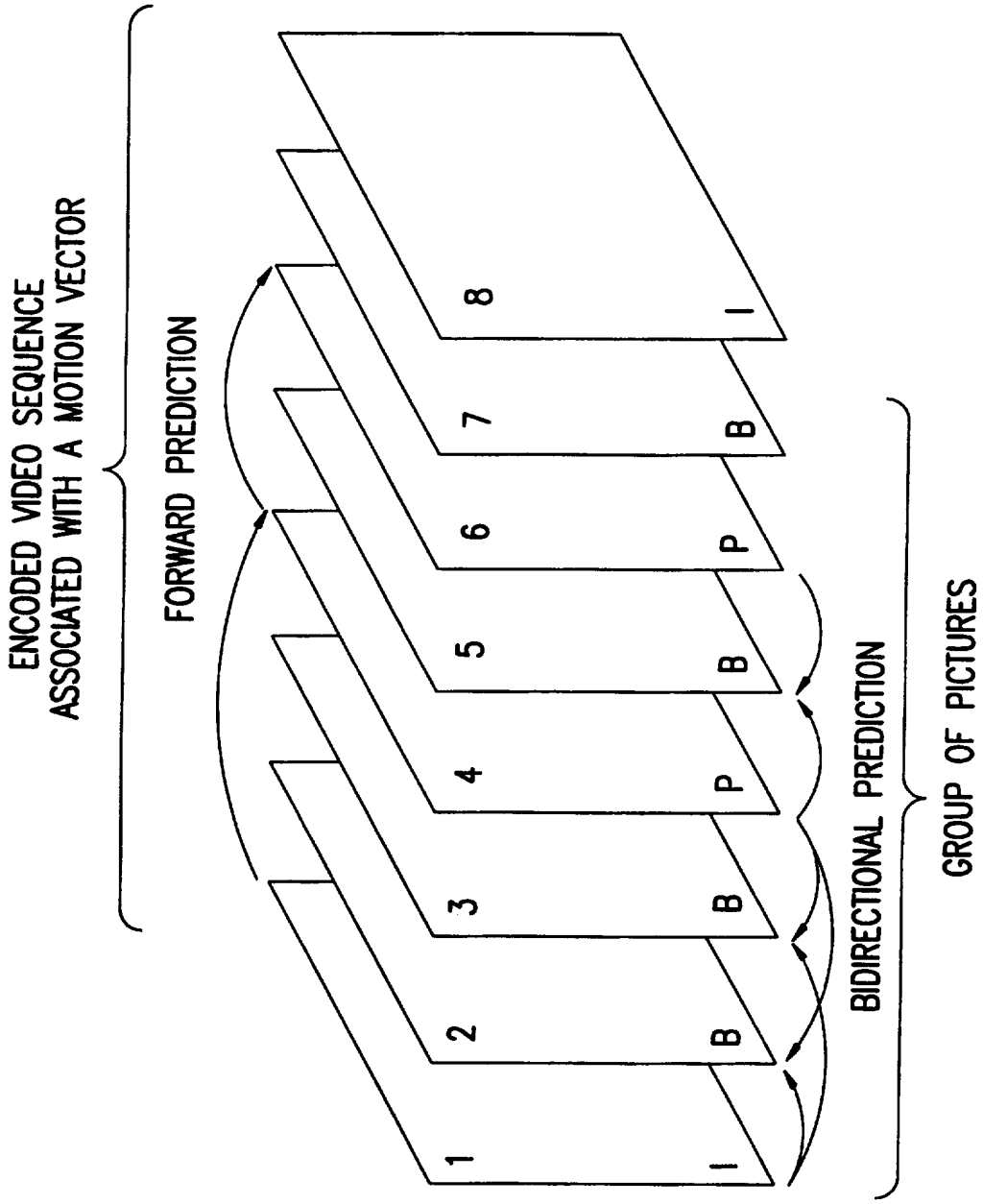

ENCODED VIDEO SEQUENCE
ASSOCIATED WITH A MOTION VECTOR
FORWARD PREDICTION
BIDIRECTIONAL PREDICTION
GROUP OF PICTURES
1
2
3
4
5
6
7
8
I
B
B
P
B
P
B
B
I

FIG.1

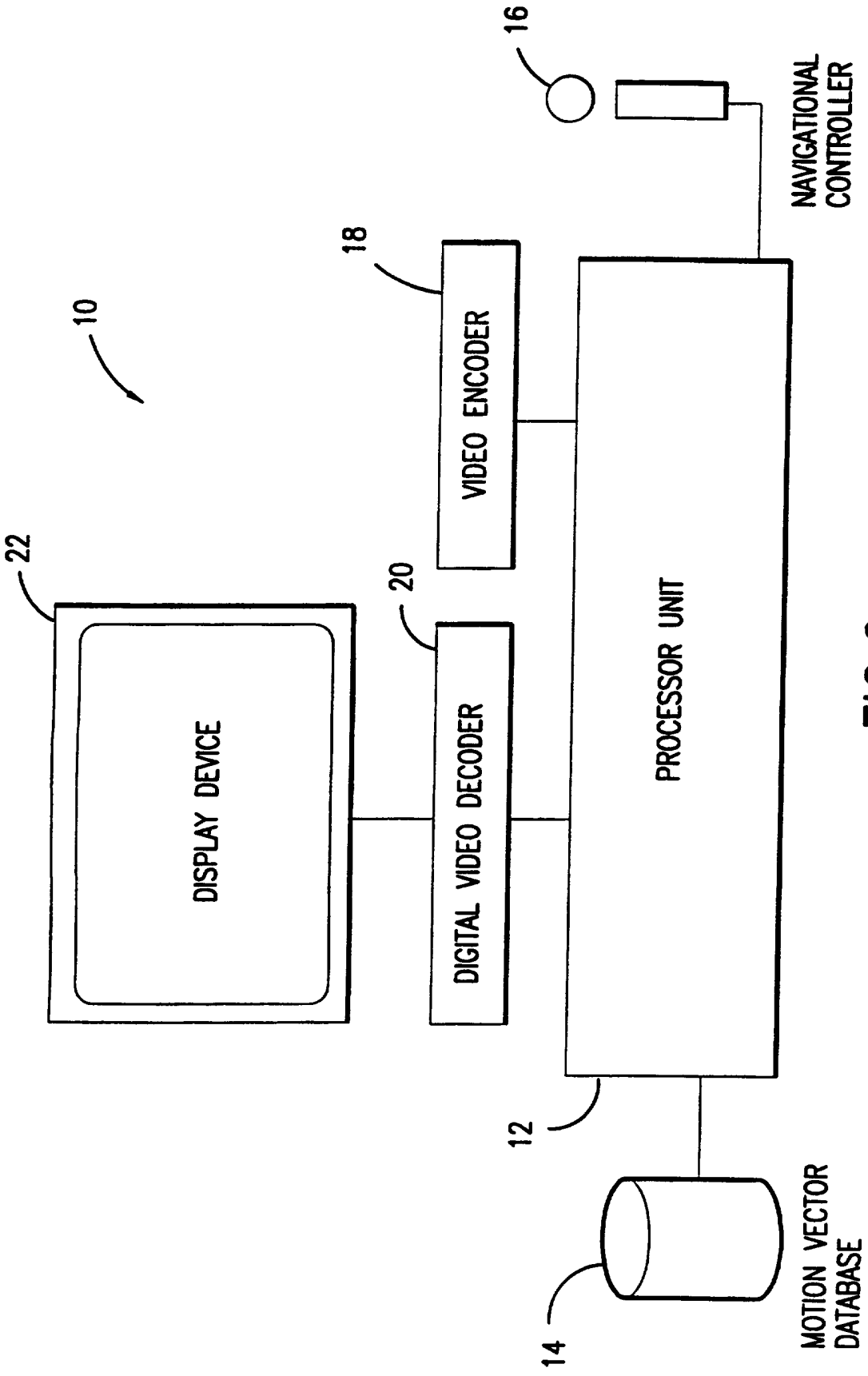

16
NAVIGATIONAL CONTROLLER
18
VIDEO ENCODER
10
22
20
DISPLAY DEVICE
DIGITAL VIDEO DECODER
PROCESSOR UNIT
12
MOTION VECTOR DATABASE
14

FIG.2

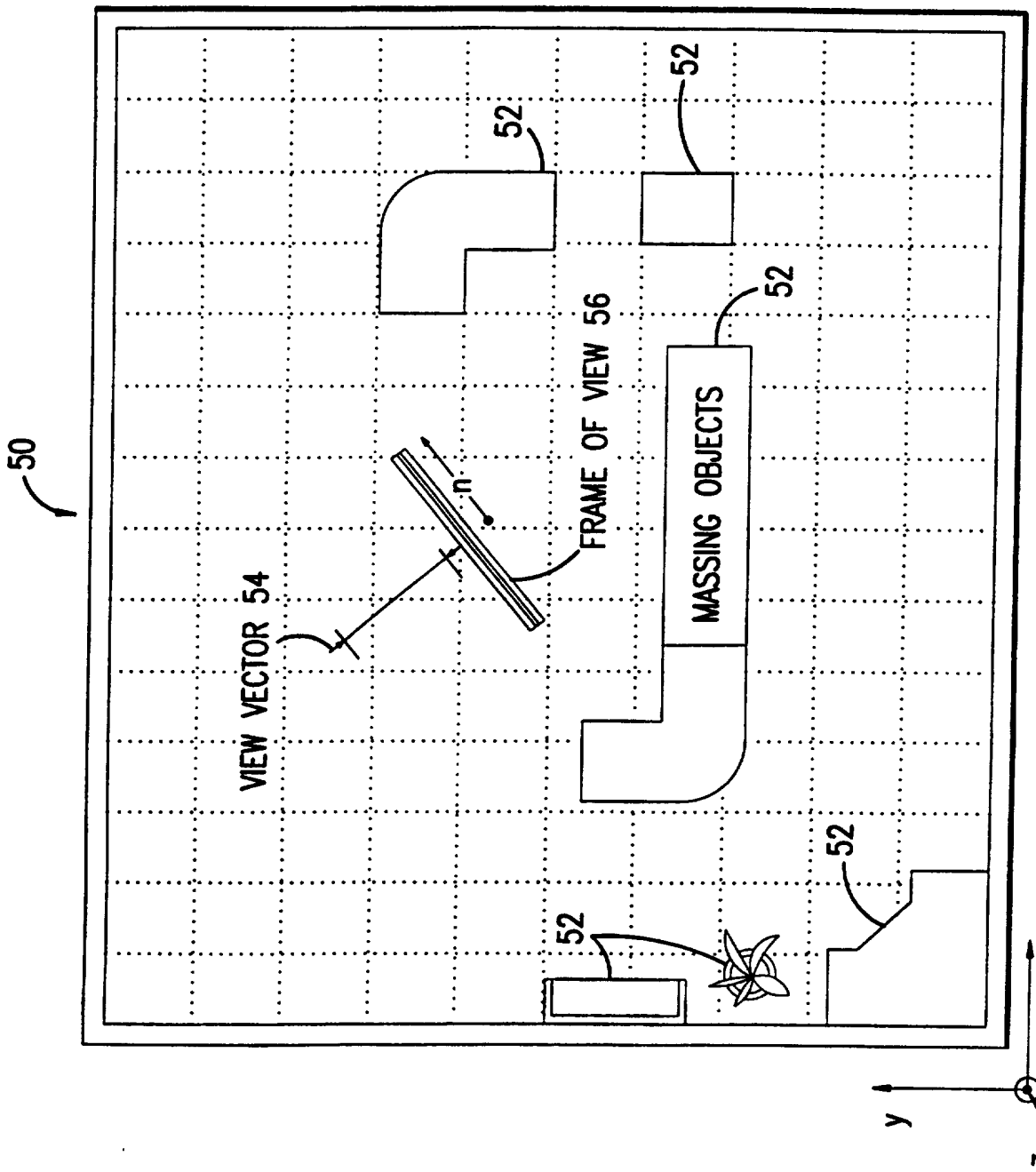

50
52
52
52
52
52
FRAME OF VIEW 56
MASSING OBJECTS
n
VIEW VECTOR 54
x
y
z

FIG.3

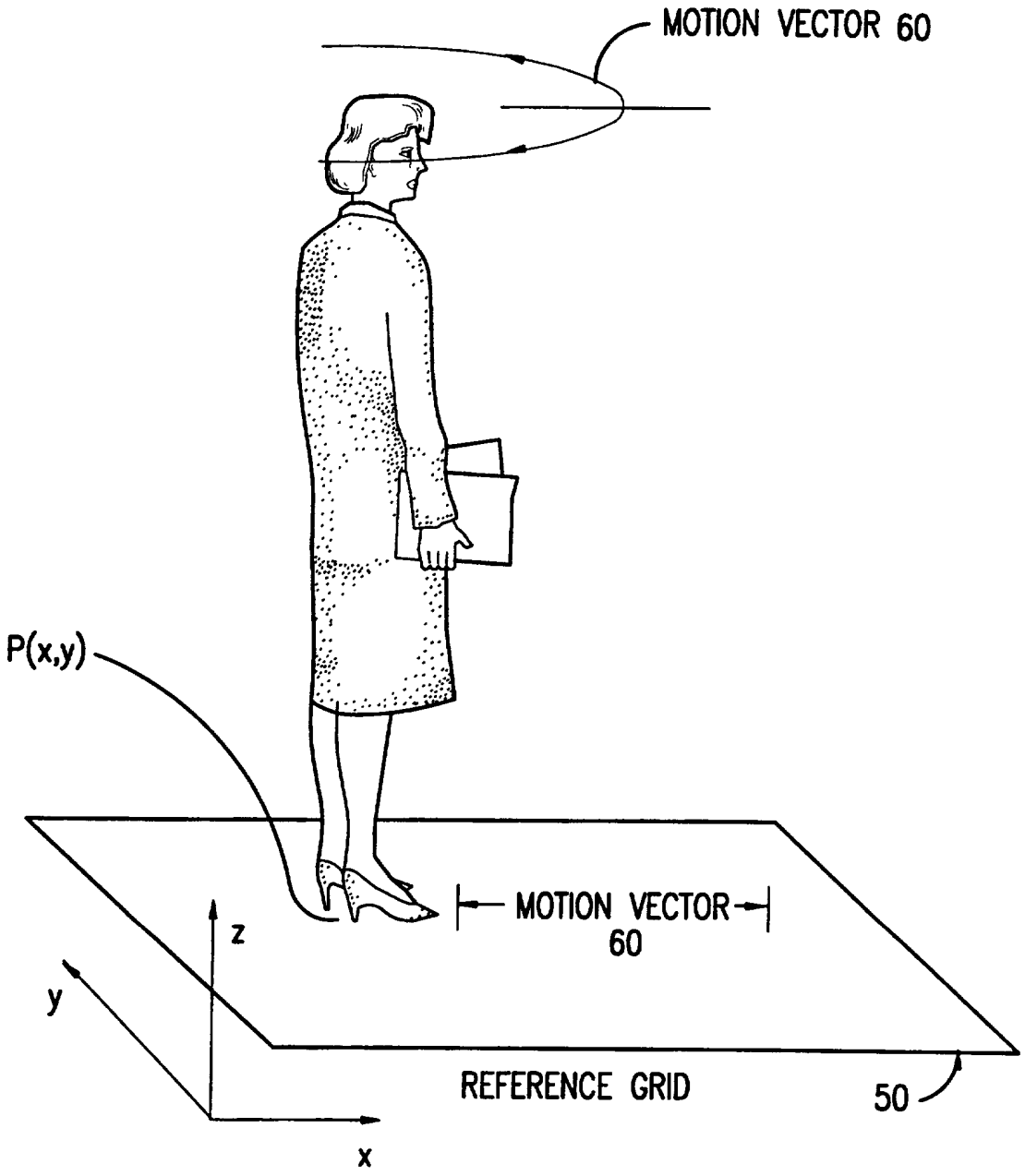


FIG.4

INSTRUCTION	CAMERA	X POSITION	Y POSITION	Z POSITION	FRAME
MOVE	DCAM2	156	24	72.00	0
MOVE	DCAM2	156	48	72.00	15
MOVE	DCAM2	156	72	72.00	30
MOVE	DCAM2	156	96	72.00	45
MOVE	DCAM2	156	120	72.00	60
MOVE	DCAM2	156	144	72.00	75
MOVE	DCAM2	156	168	72.00	90
MOVE	DCAM2	156	192	72.00	105
MOVE	DCAM2	156	216	72.00	120

FIG.5

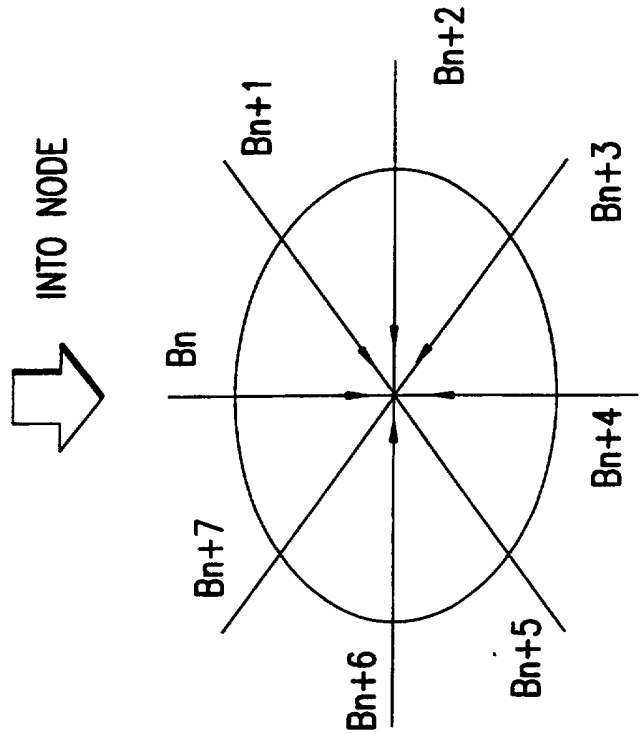

INTO NODE
Bn
Bn+1
Bn+2
Bn+3
Bn+4
Bn+5
Bn+6
Bn+7

FIG.6B

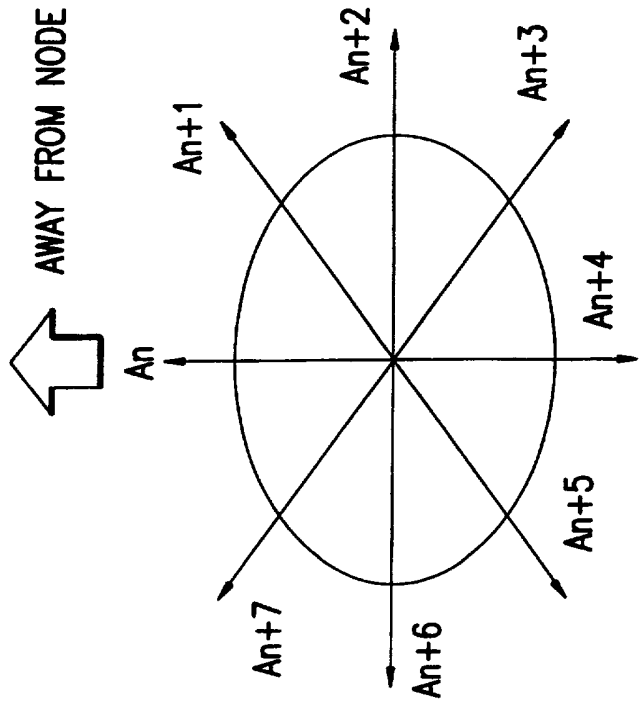

AWAY FROM NODE
An
An+1
An+2
An+3
An+4
An+5
An+6
An+7

FIG.6A

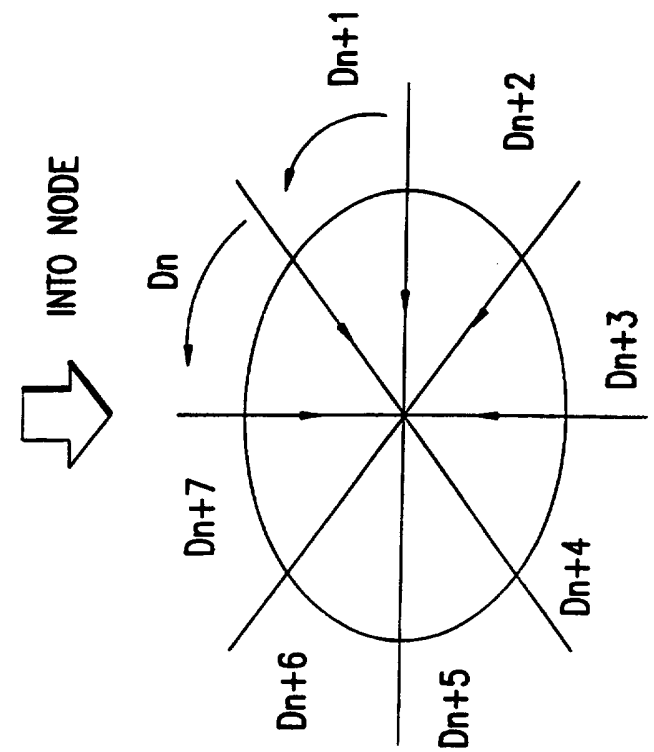

INTO NODE
Dn
Dn+1
Dn+2
Dn+3
Dn+4
Dn+5
Dn+6
Dn+7

FIG.6D

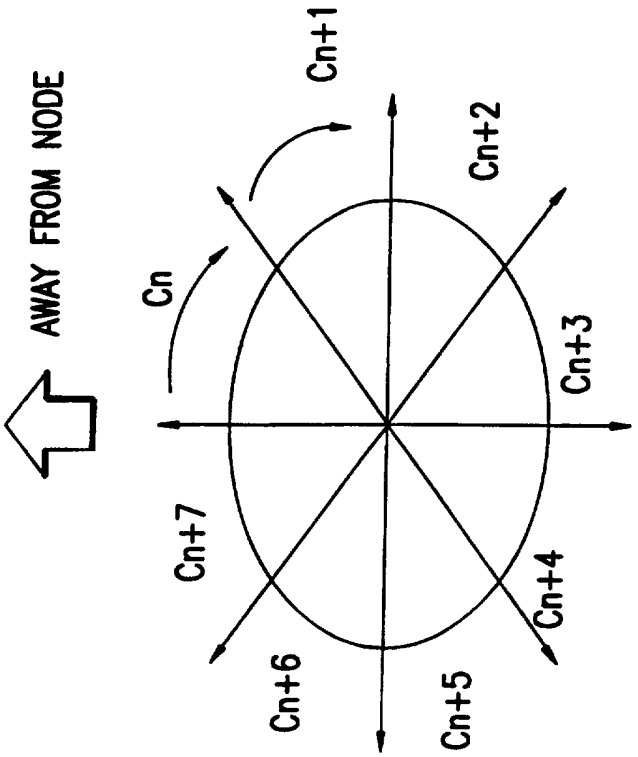

AWAY FROM NODE
Cn
Cn+1
Cn+2
Cn+3
Cn+4
Cn+5
Cn+6
Cn+7

FIG.6C

FIG.7A

STATE TABLE FOR MOVE BACKWARD SUBSTATE			
CPV	MV	NPV	NPCC
An+2	Bn+2	CPV	2
An	Bn	CPV	6
An+4	Bn+4	CPV	5
An+6	Bn+6	CPV	1
An+1	Bn+1	CPV	4
An+3	Bn+3	CPV	8
An+5	Bn+5	CPV	3
An+7	Bn+7	CPV	7

NPCC CODE	X, Y POSITIONS
0	X, Y
1	X+K,Y
2	X−K,Y
3	X+K, Y+K
4	X−K, Y−K
5	Y+K,X
6	Y−K,X
7	X+K, Y−K
8	X−K, Y+K

FIG.7B

STATE TABLE FOR MOVE FORWARD SUBSTATE			
CPV	MV	NPV	NPCC
An+2	CPV	CPV	1
An	CPV	CPV	5
An+4	CPV	CPV	6
An+7	CPV	CPV	2
An+1	CPV	CPV	3
An+3	CPV	CPV	7
An+5	CPV	CPV	4
An+7	CPV	CPV	8

KEY TO THE STATE TABLE AND TERMS:

CPV = CURRENT POSITION VECTOR

MV = MOTION VECTOR

NPV = NEXT POSITION VECTOR

NPCC = NEXT POSITION CHANGE COORDINATE

K = NORMALIZED STEPWISE FACTOR

X,Y – CARTESIAN COORDINATES OF QUADRANT

0 = NO OPERATION

FIG.7C

STATE TABLE FOR TURN RIGHT SUBSTATE			
CPV	MV	NPV	NPCC
An	Cn	An+1	0
An+1	Cn+1	An+2	0
An+2	Cn+2	An+3	0
An+3	Cn+3	An+4	0
An+4	Cn+4	An+5	0
An+5	Cn+5	An+6	0
An+6	Cn+6	An+7	0
An+7	Cn+7	An	0

NPCC CODE	X, Y POSITIONS
0	X, Y
1	X+K,Y
2	X−K,Y
3	X+K, Y+K
4	X−K, Y−K
5	Y+K,X
6	Y−K,X
7	X+K, Y−K
8	X−K, Y+K

FIG.7D

STATE TABLE FOR TURN LEFT SUBSTATE			
CPV	MV	NPV	NPCC
An	Dn+7	An+7	0
An+1	Dn	An	0
An+2	Dn+1	An+1	0
An+3	Dn+2	An+2	0
An+4	Dn+3	An+3	0
An+5	Dn+4	An+4	0
An+6	Dn+5	An+5	0
An+7	Dn+6	An+6	0

KEY TO THE STATE TABLE AND TERMS:

CPV = CURRENT POSITION VECTOR

MV = MOTION VECTOR

NPV = NEXT POSITION VECTOR

NPCC = NEXT POSITION CHANGE COORDINATE

K = NORMALIZED STEPWISE FACTOR

X,Y – CARTESIAN COORDINATES OF QUADRANT

0 = NO OPERATION

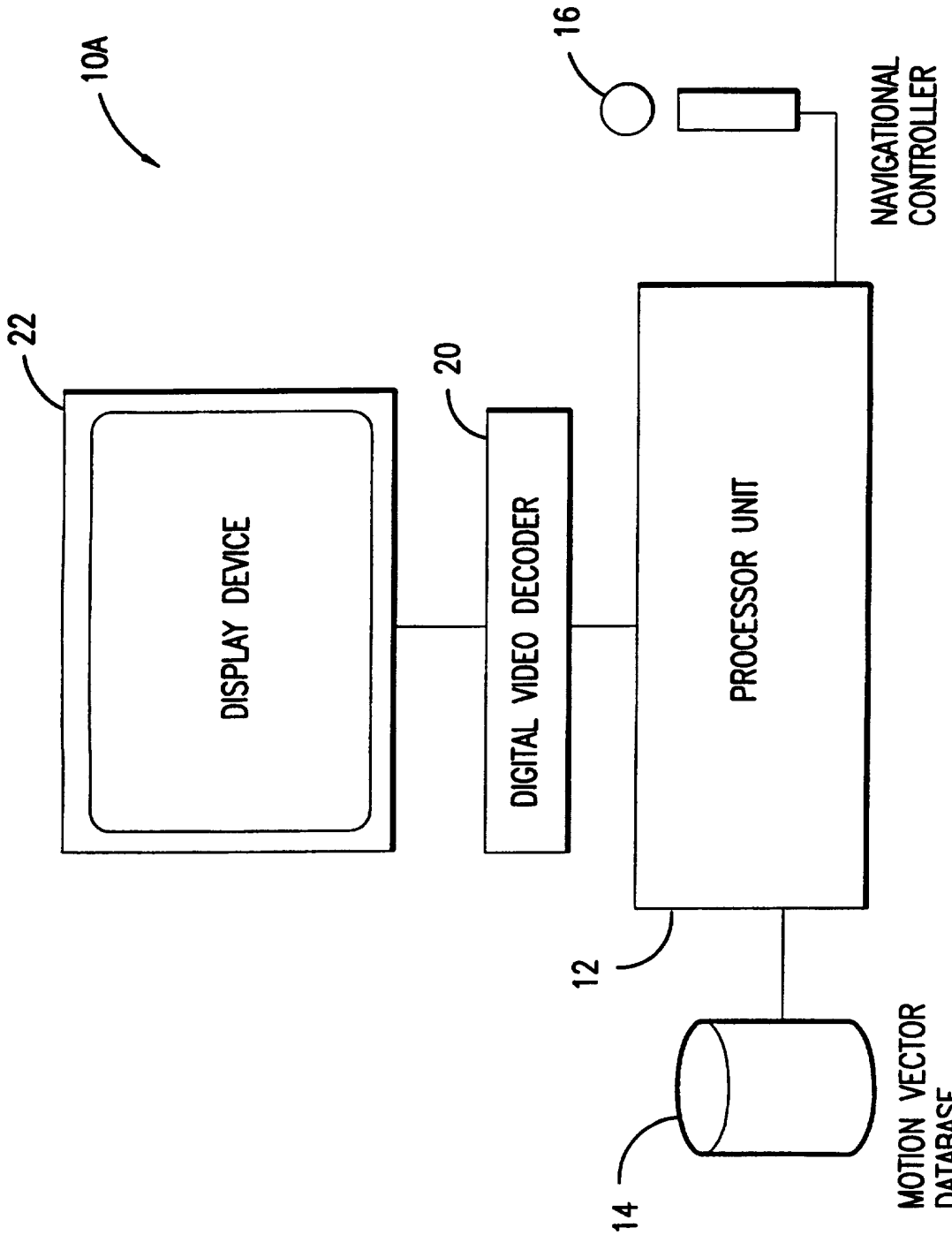

10A
16
NAVIGATIONAL CONTROLLER
22
20
DISPLAY DEVICE
DIGITAL VIDEO DECODER
PROCESSOR UNIT
12
14
MOTION VECTOR DATABASE

FIG.8

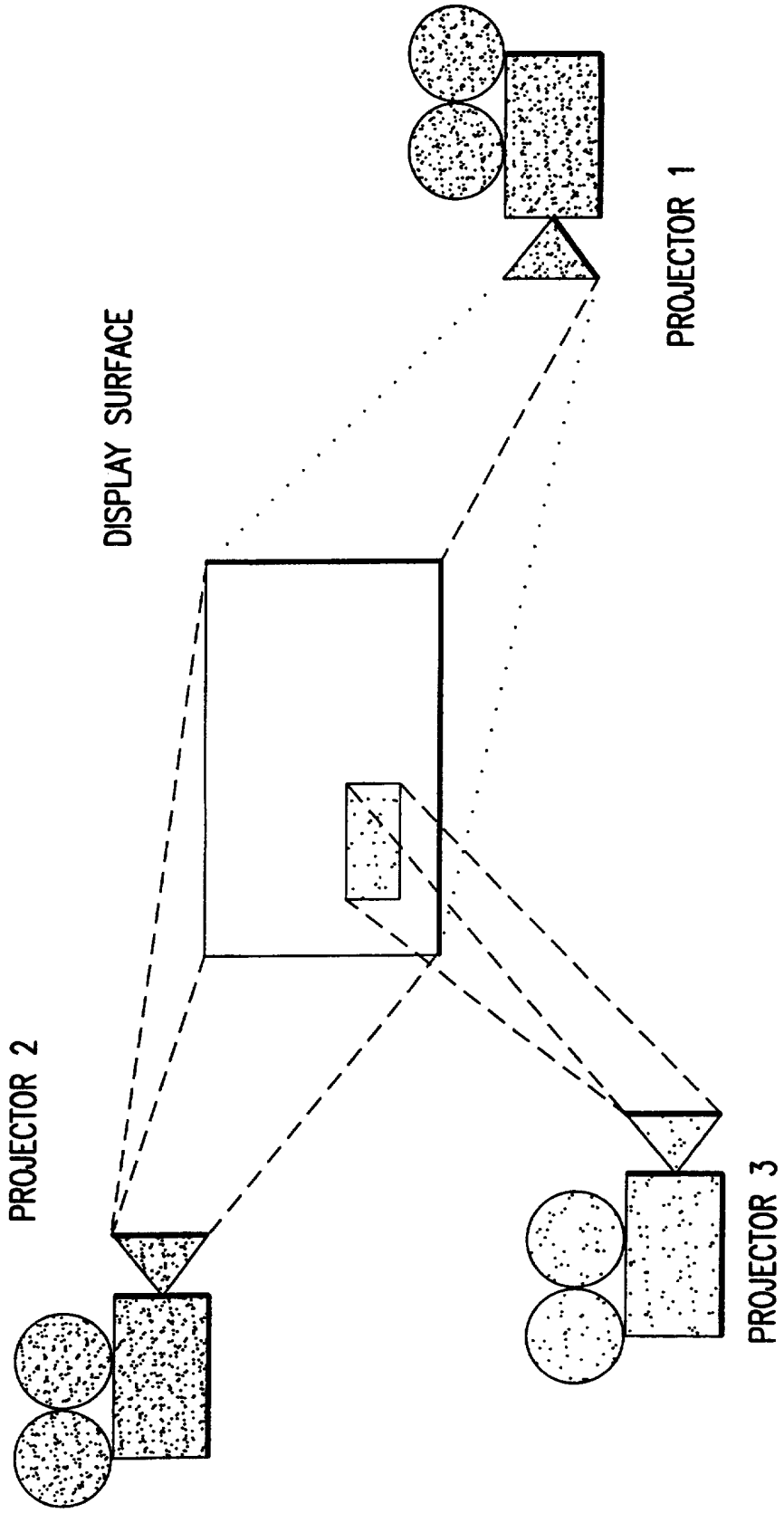


FIG.9

14/15

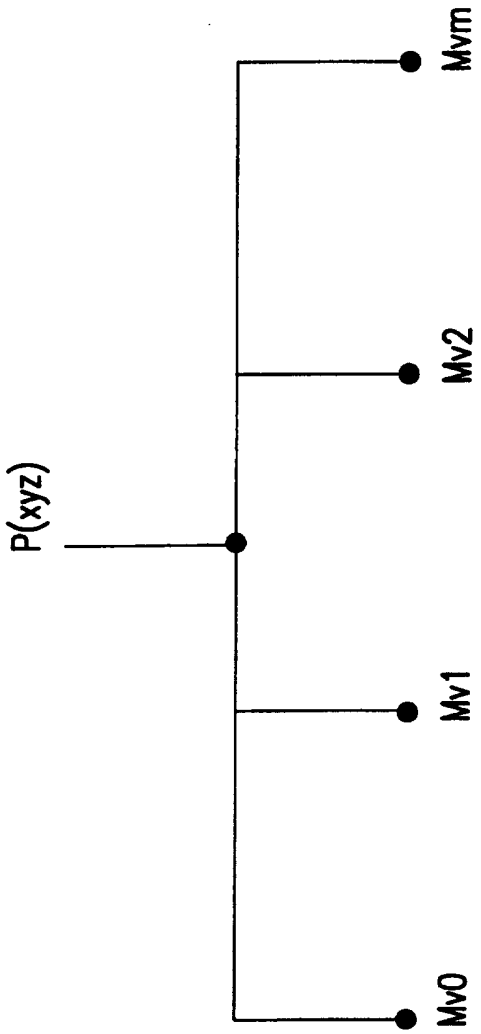
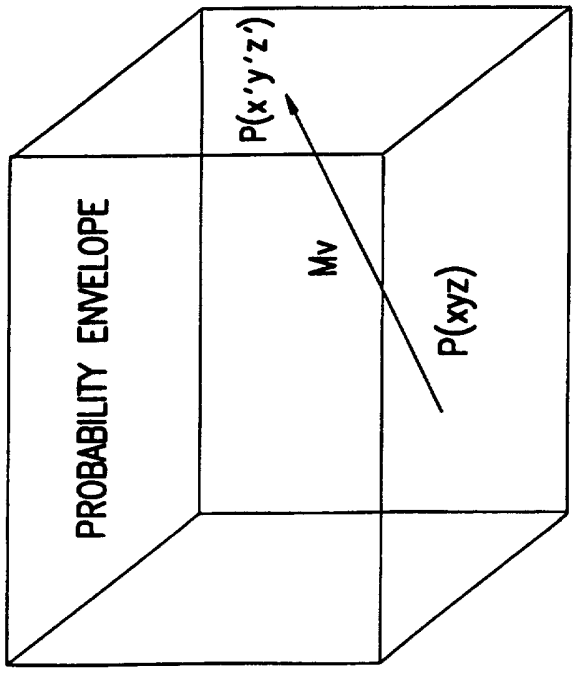


FIG.10



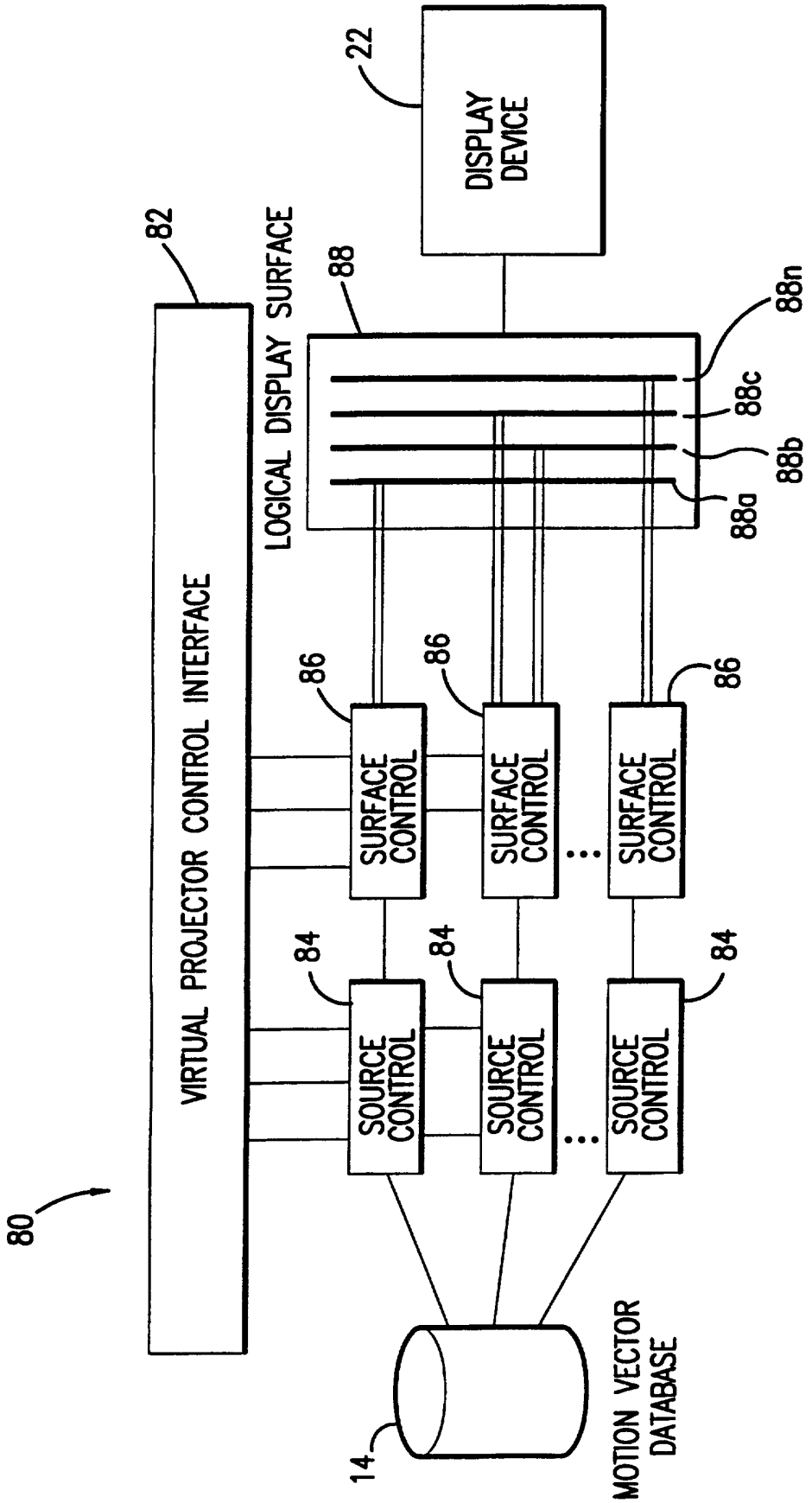

80
82
VIRTUAL PROJECTOR CONTROL INTERFACE
22
DISPLAY DEVICE
88
LOGICAL DISPLAY SURFACE
88n
88c
88b
88a
86
SURFACE CONTROL
84
SOURCE CONTROL
14
MOTION VECTOR DATABASE

FIG.11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/02290

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06T 3/00
US CL :Please See Extra Sheet.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/173, 174, 175, 806, 807, 119, 137, 138; 382/154, 236, 295, 296; 345/8, 121, 122; 348/39

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS
search terms: virtual reality, navigat?, frame?, walkthrough, MPEG, sequence? (2a) image?

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,287,437 A (DEERING) 15 February 1994, col.2, line 46; col.4, line 58; col.5, line 63 to col.6, line 19; col.7, line 53; col.8, lines 7-47; col.11, line 23; col.12, lines 31-64; col.13, lines 4, 23.	1-3
A, P	US 5,499,146 A (DONAHUE et al) 12 March 1996.	1-3
A, E	US 5,615,132 A (HORTON et al) 25 March 1997.	1-3
A, E	US 5,617,334 A (TSENG et al) 01 April 1997.	1-3
A, P	US 5,577,981 A (JARVIK) 25 November 1996.	1-3
A	US 5,130.794 A (RITCHEY) 14 July 1992.	1-3
A	US 5,388,990 A (BECKMAN) 14 February 1995.	1-3

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 02 APRIL 1997	Date of mailing of the international search report 17 APR 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer STEPHEN HONG Telephone No. (703) 308-5465

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/02290

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A, P	US 5,577,961 A (ADAMCZYK et al) 26 November 1996.	1-3
A, P	US 5,579,026 A (TABATA) 26 November 1996.	1-3
A, P	US 5,581,271 A (KRAEMER) 03 December 1996.	1-3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/02290

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

395/173, 174, 175, 806, 807, 119, 137, 138; 382/154, 236, 295, 296; 345/8, 121, 122; 348/39