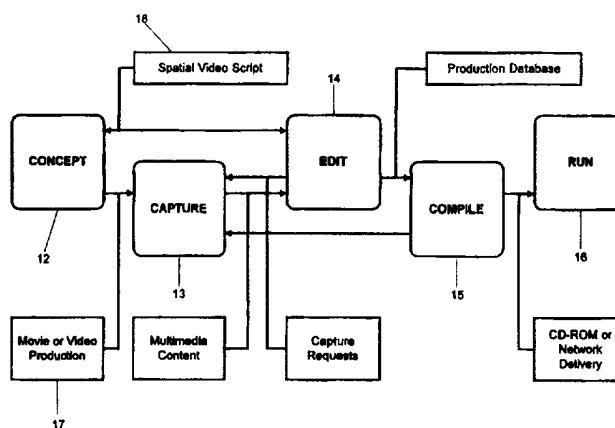




## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification <sup>6</sup> : <b>G06T 5/10</b>		<b>A1</b>	(11) International Publication Number: <b>WO 97/42601</b>
			(43) International Publication Date: 13 November 1997 (13.11.97)
(21) International Application Number: PCT/US97/07359 (22) International Filing Date: 2 May 1997 (02.05.97) (30) Priority Data: 60/016,975 6 May 1996 (06.05.96) US (71) Applicant: SAS INSTITUTE, INC. [US/US]; SAS Campus Drive, Cary, NC 27512 (US). (72) Inventors: BAKER, David, F.; 6513 Johnsdale Road, Raleigh, NC 27615 (US). HALES, William, C.; P.O. Box 130, Apex, NC 27502 (US). KAISER, Paul, A.; 2218 Walden Creek Drive, Apex, NC 27502 (US). PACE, Jason, K.; 7808 Highlandview Circle, Raleigh, NC 27613 (US). PERRIN, Stephen, R.; 1706 Michaux Road, Chapel Hill, NC 27514 (US). TOEBES, John, A., VIII; 207 Livingstone Drive, Cary, NC 27513 (US). WALKER, Douglas, J.; 4701 Oak Park Road, Raleigh, NC 27612 (US). (74) Agents: EVANS, Richard, W. et al.; Moore & Van Allen, PLLC, Suite 800, 2200 West Main Street, Durham, NC 27705 (US).		(81) Designated States: European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).  <b>Published</b> <i>With international search report.</i> <i>Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>	

(54) Title: INTEGRATED INTERACTIVE MULTIMEDIA PROCESS



Spatial Video Production Process

## (57) Abstract

A navigable multimedia system displaying a navigable motion video (16) based representation of an environment having paths, wherein at least one clip is associated with each path, and allowing for spontaneous navigation of the paths in response to user input. The system provides for seamless transition from a video clip of one path to one of an intersecting path and from one field of view along a path to another field of view on that path. The system provides for hotspots associated with features in the environment and bitmap representations of objects to be placed in receptacles associated with certain portions of the environment wherein said hotspots and receptacles remain associated with environmental features and wherein said objects appear in proper perspective regardless of viewer position. A method for creating a navigable motion video environment comprising the steps of capturing (13) in video clips having optical properties an environment field of view along natural paths through the environment, capturing M-view sprites and other elements and correlating with map clips, adding and editing hotspots, receptacles, and other multimedia production components and binding into final production.

**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.

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

## INTEGRATED INTERACTIVE MULTIMEDIA PROCESS

### CROSS-REFERENCES

This application claims the benefit of U.S. Provisional Application No. 60/016,975,  
5 filed May 6, 1996.

### COPYRIGHT AUTHORIZATION

A portion of the disclosure of this patent document contains material which is subject  
to copyright protection. The copyright owner has no objection to the facsimile reproduction  
10 by anyone of the patent document or the patent disclosure, once such are publicly available,  
but otherwise reserves all copyrights whatsoever.

### FIELD OF THE INVENTION

This invention relates to an improved process for the creation of interactive  
15 multimedia environments, called Productions, which use video annotated with position  
information in the creation and navigation of a virtual reality environment; and more  
particularly, it incorporates all stages of development of a Production from concept inception  
to the final delivered product.

### BACKGROUND OF THE INVENTION

Interactive multimedia is a vague term which generally refers to a production  
playable on a computer in which multiple types of media are present, possibly including text,  
sound, graphics, animation and video. Such productions are contrasted to older style  
computer programs which are limited to text or to text and graphics. Hence the  
25 "multimedia" component of the name. Further, such productions allow for the course of the  
production to be affected or controlled to some extent by the user, hence the "interactive"  
component of the name.

Products claiming to be interactive multimedia products may contain widely varying  
degrees of interactivity and multimedia. On one extreme a production may be full of sound,  
30 still pictures, animation and other video clips but offer the user very little in the way of

interaction or control. On another extreme the production may be highly interactive with the outcome and course of the production greatly affected by user interaction but have little to offer in the way of video, animation or sound.

It has been said that there are as many definitions of virtual reality as there are people who have heard of it. For the purposes of this disclosure virtual reality describes a subset of interactive multimedia productions that let the user navigate and interact with a three-dimensional computer-generated environment in real time. Such productions have three defining elements: interaction, 3-D graphics, and immersion. 3-D graphics lets the user see the environment. Real time updates to the graphics being displayed in reaction to the user's changing point of view contribute to the feeling of presence or immersion in the environment. The degree of immersion depends on the production. Just as the case with interactive multimedia, there is a range of virtual reality. Some systems with motion sensing video helmets and glove input devices have a higher degree of immersion than 3-D productions where the 3-D graphics are displayed on a computer monitor and input is through a keyboard or mouse.

In computer rendered virtual reality environments, real-time updates of the computer rendered graphic scenes in reaction to the user's changing point of view help to create a feeling of presence in the virtual world. Such "on-the-fly" rendered virtual reality worlds allow the highest degree of user navigation and interactivity, providing almost unlimited, on the fly navigation to the user, even if the user chooses to move in unusual or unnatural ways through the environment. Such navigation requires the computer to constantly recalculate the object angles, viewing perspectives and the like in the environment and redraw the display with the updated information as the user navigates. These on-the-fly updates are very computationally demanding, thus, computer rendered virtual worlds which can be generated in real time on a home computer, such as that used on the DOOM™ video game must necessarily have limited resolution and a limited visual complexity or they will suffer from unacceptably slow performance.

The high degree of navigation and interaction offered by true virtual reality as described above and the limited quality of the resolution of the graphics of the virtual world contrast with the other end of the continuum which is linear video. Linear video played on a computer has very high resolution but no on-the-fly navigation. It is employed in interactive



multimedia productions under circumstances where little interaction is needed. Just as interactive video becomes more appealing as the user's sense of "being there" is enhanced, appeal is further enhanced as the user's sense of control is increased. Factors such as the level of interactivity and the frequency of interactivity are often discussed as influencing such things as the length of video clips to be used in a production. Frequent interaction, as in video games where there can be more than 100 interactions per minute by each player, is said to call for short video segments, if any. On the other hand, use of video in information kiosks may involve as few as 1 or 2 interactions per minute, thus permitting video clips to be much longer. In situations where user interaction can be virtually continuous, as in many of the currently popular racing/flying video games, the scenes presented are invariably rendered scenes. This is true whether such rendering is done ahead-of-time or in real-time.

One attempt to obtain an immersion-type environment with greater visual resolution than currently available with the on-the-fly rendered environments of true virtual reality is found in those technologies which create a 3-D world from captured real-world still pictures which are converted into digitized panoramic bitmaps in the computer; such technologies include QuickTime VR from Apple and Surround Video from Microsoft. Panoramic video technology uses digitized panoramic photos of real-world scenes to create a rich visual environment. Despite their names, QuickTime VR and Surround Video employ still pictures or stationary video images rather than motion video footage to create such panoramas. The term "motion video", as used herein, refers to a series of images captured in rapid succession so that the illusion of continuity and motion is created.

Panoramic video uses computer-digitized and stored photos of real-world scenes which have been mapped into 360-degree panoramas enabling them to be presented on a 2-D screen without the optical distortion so common with photographic techniques. We refer to these technologies as panoramic video. Panoramic video technology does not permit on-the-fly rendering of the environment. Rather, the user is permitted some degree of navigation within the previously stored panorama because the panoramic picture captures much more video information than can be placed on the display at one time. In response to user input, the computer can pan around the axis of the panorama, allowing the user a 360-degree turning motion in the horizontal plane. Depending on the viewer characteristics and the vertical size of the panorama, the viewer may be allowed some vertical panning as well.

Furthermore, the developer of the panoramic video production may create some points in each panorama where the user can "jump" or transition to the next panorama.

Panoramic video techniques create a sense of presence in the environment by allowing users to control, within limits, their point of view, to navigate to varying degrees, and to interact with a scene or environment perceived to be three-dimensional although presented on a two-dimensional computer or video display device. Panoramic video technologies also allow for the incorporation of rendered scenes, but do not use the traditional virtual reality technique of computing the environment in real time in response to the user's changing perspective. Instead they allow the user some degree of navigation by panning the display left or right around the panorama in a complete circle, zooming in and out on user-selected areas of the displayed panorama. Productions made with panoramic video techniques may even allow the user to shift position to other view points at predefined hotspots where the developer has linked panoramas. As in computer generated virtual environments, panoramic video allows the user to interact with predefined hotspots which may generate dramatic video or audio responses and the like.

The panoramic video technology also includes the ability to define hotspots on the panoramic and link such hotspots to views of an object. For example, if a statue is in a defined hotspot and that hotspot is clicked on, the activated hotspot can cause a special window to open on the screen, if the producer has incorporated it into the work, the window may display an enlarged picture of the object which image can be rotated 360 degrees horizontally and almost 360 degrees vertically. However the object being examined is removed from the environment when it is examined and its degree of horizontal and vertical rotation is not related to the perspective of the viewer in the panoramic environment.

*Surround Video* has the additional capability of allowing the multimedia maker to overlay scenes which have been shot against a "blue screen" on top of the panoramic video such as an overlaid video image of a narrator to a tour of the scene presented by the panorama. However, Surround Video has no capability of changing the scene in response to movement of the viewer in the panoramic environment.

Despite the advantages of panoramic video, its limitations limit its use in highly interactive productions. For example, user viewpoints, and transitions between viewpoints in panoramic video, are limited to those that have been previously defined by the developer

and may only be to or from a position at which the currently viewed panorama or another panorama has been produced. Further, due to cost and storage issues it is impractical to generate a different panoramic video for every foot of space along the viewer's navigation paths through the environment. This limitation can introduce a perceived navigational discontinuity of the viewed environment and can detract from the feeling of presence in that environment. Panoramas are very large images and require a significant amount of computer storage space. Further, the rotatable objects in QuickTime VR must capture the screen to be viewed and, therefore, are not integrated into the environment in a way which enhances the viewer's immersion of the environment. In QuickTime VR there is no way of adding objects into the environment which were not there when it was filmed, or removing objects from the environment which were there when filmed (other than simply filming the environment both with and without the object and replacing one panorama with the other at some point in the production). This limits the viewer's ability to interact with the environment. Moreover the rotatable objects are also storage intensive as they may require 500 views of the object to create. Finally, while the blue screen techniques available in Surround Video, which are likely to be available in later versions of QuickTime VR, allow some integration of objects into the environment, there is no simple way provided for the object to be examined by the viewer from various perspectives.

Panoramic video has an advantage over rendered virtual reality systems in that storage and retrieval of pre-created panoramic images is less computationally demanding than on the fly recalculation of rendered scenes. Thus scenes which are not practical to render can be photographed. However, use of digitized pictures with high quality resolution and color places increased demands on the computer's storage capacity and transfer rates. Further, the sense of immersion provided by panoramic video suffers in comparison to true virtual reality productions because of its significantly lower degree of navigational freedom. Further, in panoramic video it is difficult to create the impression that one is moving through an environment which in any way responds to the viewer's movement. For example, it would be difficult to shoot panoramic video where the user was going down a jungle path and, as the user passed a branch hanging out into the user's path, the user pushed the branch out of the way.

Despite advances in the art, there remain serious limitations in personal computer performance which create difficulties for developers seeking to create realistic, navigable computer environments. Among those are the amount of data required to create rich environments. The amount of data required can severely tax storage means. Further, for an environment to be truly interactive, the data on the screen must be able to change in response to the user input. Often productions created using the current state of the art must pause while the computer seeks the desired data and while the computer draws the new environment for display, thus the seek time limitations of the presently available storage means, such as the CD-ROM creates situations which detract from the sense of immersion or reality to which the user is experienced due to pauses during seek times. Further, rich graphical environments require more time to construct, further use of motion video also requires substantial transfers of data, places significant demands on the bandwidth limitations of the storage media and the processor.

Advances in processor speed, larger and faster access memory, caching techniques using memory with even faster read-write cycle times, increased hard disk drive capacities at affordable prices, and higher transfer rates from CD-ROM disk drives have all contributed toward meeting the computational and storage demands of interactive multimedia applications. Yet, limitations in processor speed, and storage capacity, access time and transfer rates forces compromises in creating a rich visual environment on a home computer, rich in objects with which the user can interact and with a high degree of navigability. The present invention which we refer to as spatial video or Video Reality™ and the spatial video production and runtime systems overcome these limitations in a novel way.

While it can be used without such compression, in order to reduce storage requirements to the level supported by home computer technology, the present invention employs video digital compression technology to reduce storage demands. Before an analog video signal can be processed by a computer it has to be digitized. NTSC video has 480 lines per video frame and 640 pixels per line. Working with 8-bit color, then requires 9.2 MB of raw data to represent one second of non-compressed video; 24-bit color requires 27.6 MB of raw data per second of video. A one-hour video presentation would require 99.5 GB.

With such high storage rates, it was imperative that the multimedia PC/CD-ROM and related industries adopt a digital video compression and decompression standard.

A number of compression standards have been employed in the prior art using various mathematical algorithms to achieve impressive degrees of compression while maintaining good picture quality. The methods and devices of the current invention are designed to work with any digital compression technologies known to the inventor.

5 One such standard which has gained appreciable acceptance is MPEG-1. The MPEG-1 standard is found in the publication "International Standard ISO/IEC 11172, "Information technology - coding of motion pictures and associated audio for digital storage media at up to about 1.4 Mbits/s - Part 2: Video" which is incorporated herein by reference. The MPEG-1 standard is discussed in inventor's co-pending U.S. patent applications filed on  
10 February 19, 1997 as serial numbers 08/802,870 and 08/801,254, which are incorporated herein by reference and are referred to herein as the "Special Effects Applications."

Since the adoption of the MPEG standard, there has been a plethora of hardware and software tools for the personal computer to assist video and multimedia authors, producers, and users in the authoring, capturing, editing, compiling/binding and running or playing of  
15 multimedia productions. The present invention will make use of these tools wherever practical in the process which will be defined.

#### Overview of the MPEG-1 Algorithm

The coded representation defined in Part 2 of MPEG-1 achieves a high compression  
20 ratio while preserving good picture quality. The algorithm is not lossless as the exact pel values are not preserved during coding. The choice of techniques is based on the need to balance a high picture quality and compression ratio with the requirement to make possible random access to the coded bit stream.

A number of techniques are used to achieve a high compression ratio. The first is to  
25 select an appropriate spatial resolution for the signal. The algorithm then uses block-based motion compensation to reduce the temporal redundancy. Motion compensation is used for causal prediction of the current picture from a previous picture, for non-causal prediction of the current picture from a future picture, or for interpolative prediction from past and future pictures. Motion vectors are defined for each 16-pel by 16-pel region of the picture. The  
30 difference signal, the prediction error, is further compressed using a discrete cosine transform (DCT) to remove spatial correlation before it is quantized in an irreversible

process that discards the less important information. Finally, the motion vectors are combined with the DCT information, and coded using variable length codes.

MPEG-1 uses three main picture types. Intra-coded pictures (I-Pictures) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only a moderate compression ratio. Predictive coded pictures (P-Pictures) are coded more efficiently using motion compensated prediction from a past intra or predictive coded picture and are generally used as a reference for further prediction. Bidirectionally-predictive coded pictures (B-Pictures) provide the highest degree of compression but require both past and future reference pictures for motion compensation.

Bidirectionally-predictive coded pictures are never used as reference for prediction. The organization of the three picture types in a sequence is very flexible. The choice is left to the encoder and will depend on the requirements of the application. The fourth picture type, the D-picture is provided to allow a simple, but limited quality, fast-forward playback mode.

Structure of the Coded Video Bit Stream. Part 2 (relative to video) of the MPEG-1 standard specifies a syntax for a coded video bit stream. This syntax contains six layers, each of which either supports a signal processing or a system function; the layers are hierarchical with the highest layer being the broadest in scope:

<i>Sequence Layer</i>	<i>Context Random Access Unit</i>
<i>Group of Pictures Layer</i>	<i>Video Random Access Unit</i>
<i>Picture Layer</i>	<i>Primary Coding Unit</i>
<i>Slice Layer</i>	<i>Resynchronization Unit</i>
<i>Macroblock Layer</i>	<i>Motion Compensation Unit</i>
<i>Block Layer</i>	<i>DCT Unit</i>

The MPEG-1 video bit stream syntax follows the hierarchical structure of the layers discussed above. A sequence is the top level of video coding. It begins with a sequence header which defines important parameters needed by the decoder. The sequence header is followed by one or more groups of pictures. Groups of pictures, as the name suggests, consist of one or more individual pictures. The sequence may contain additional sequence headers. A sequence is terminated by a sequence end code. MPEG-1 allows considerable flexibility in specifying application parameters such as bit rate, picture rate, picture resolution, and picture aspect ratio. These parameters are specified in the sequence header.

Following the *Sequence Layer* is the *Group of Pictures Layer*. Two distinct picture orderings exist, the display order and the bit stream order (as they appear in the video bit stream). A group of pictures is a set of pictures which are contiguous in display order. A group of pictures must contain at least one I-picture. This required picture may be followed  
5 by any number of I and P-pictures. Any number of B-pictures may be interspersed between each pair of I or P-pictures, and may also precede the first I-picture. These groups of pictures possess these particular properties:

*Property 1.* A group of pictures, in bitstream order, must start with an I-picture and may be followed by any number of I, P, or B-pictures in any order.

10 *Property 2.* A group of pictures must begin, in display order, with an I or a B-picture, and must end with an I or a P-picture. The smallest group of pictures consists of a single I-picture whereas the largest size is unlimited.

The original concept of a group of pictures was a set of pictures that could be coded and displayed independently of any other group. In the final version of MPEG-1 this is not  
15 always true, and any B-pictures preceding (in display order) the first I-picture in a group may require the last picture in the previous group in order to be decoded. Nevertheless, encoders can still construct groups of pictures which are independent of one another. One way to do this is to omit any B-pictures preceding the first I-picture. Another way is to allow such B-pictures, but to code them using only forward motion compensation.

20 *Property 3.* From a coding point of view, a group of pictures always begins with a group of pictures header, and either ends at the next group of pictures header or at the next sequence header or at the end of a sequence, whichever comes first.

Following a Group of Pictures Start Code (a defined code of 32 bits length and byte aligned), the GOP header contains a Time Code. This encodes the same information as the  
25 SMPTE time code discussed below.

The time code can be broken down into six fields:

FIELD	BITS	VALUES
Drop Frame Flag	1	
Hours	5	0 to 23
Minutes	6	0 to 59
Fixed	1	1
Seconds	6	0 to 59
Picture Number	6	0 to 60

The Time Code refers to the first picture in the GOP in display order (as opposed to bitstream order). The SMPTE time code is included in the MPEG-1 standard to provide a video time identification to applications. Depending on the encoder set up employed, the SMPTE time code present in an MPEG video stream may be created by the encoder or copied into the stream from the previously existing time code associated with the corresponding analog video frame before its digitizing and compression. The time code in the MPEG-1 data stream is used in the present invention to correlate camera tracking data with the resulting compressed digital video frames.

#### Use of Time Codes in Other Compression Algorithms and in the Television and Motion Picture Industries

Either the SMPTE or another time code format is incorporated into many other video compression formats. Further, time codes are well known in motion picture and video production. Generally time codes are electronic signals recorded on film or video tape. In video the time code is synchronized to the accompanying video signal. The purpose of time code is to uniquely identify each frame of video on a video tape or other video recording medium. This is done by assigning a number to each frame of video in an HOURS: MINUTES: SECONDS: FRAMES format.

Time codes are accurately phase-locked to the video signals with which the codes are to be used. This is necessary to insure that each time code frame is properly timed with respect to the video frame it identifies.



Production Authoring and Editing Tools.

Given the above described background, it naturally follows that authoring and editing tools for productions having restricted navigation capabilities have, up until the present invention, been mostly time-based or, in limited cases, logic-based. These tools may be further classified as being text based or visual based. Text based authoring systems are more directed to usage by a skilled programmer whereas the visual based systems, following the paradigm so prominent in the Microsoft Windows™ Operating Systems for PCs, are more directed to the non-programmer professional.

Map-based authoring and editing tools are commonly found in a genre of computer games having a maze metaphor. The general purpose of such map editors is to allow the user (meaning, as used here, the game developer) to “draw” the world of the game from a top-down view and to place “objects” within that world. As used in this context, “objects” can be walls, doors, monsters, foods, potions, or any other item that may be useful in creating a fantasy venue for the game being authored. There is no provision in these prior art map-based authoring tools for representing a correlation between a location on the map and a particular frame of video in a particular clip of video such as will be found in the present invention.

The overall process of digital editing and the use of digital non-linear editors in the form of software applications on computers is well known by those skilled in the art and will not be dealt with here extensively. While such time or logic based tools, whether script or visual, remain useful in the process described by this invention, they are deficient in that they make little or no provisions for accommodating random or continuous navigation of an environment, (about which, more will be explained later) or for linking camera position and angle with a given video frame. Thus, there is an unmet need for viewing and editing tools which have a topological and/or spatial orientation.

As mentioned earlier, the prior art teaches that the amount of motion video in a production is inversely proportional to the degree of interactivity of the production. It is against this trend that the present invention is directed; that is, a production made by the process of the preferred embodiment of the present invention will have mostly video as the source media but will still have a high degree of user interaction. While there is an abundance of tools and techniques for rendering of images and editing suites for creating

digital video productions, these tools do not provide for editing of continuous, or closed loop environments. We conclude that there is a strong need for an editing tool based on accurate spatial relationships. Such a tool should permit an accurate correlation between map position and specific frames of video scenes not found in present editing tools.

5

### Hotspots

Hotspots are but one of numerous ways to permit user participation in an interactive multimedia production. Hotspots allow the user to interact with objects and locations in the environment. As contrasted with receptacles which are discussed below, a hotspot does not  
10 require a graphical overlay on the scene. A hotspot is a defined location in a video frame. That is a location already present in the scene with some feature of interest to the player. A hotspot is usually defined by a human video editor or programmer who describes the location of the relevant feature as an area of the screen, where, upon acting thereon with a pointing device, such as clicking with a mouse, there is initiated the occurrence of a  
15 programmed action. The programmed action can be anything that can be caused to happen by a computer program. One example, is the appearance of a balloon comment as a graphic on the viewing screen. Another example is the occurrence of an audio sound when the viewer/player clicks a pointing device on a hotspot. Hotspots have x and y dimensions and, in technologies preceding the current invention, a location on the screen that remained in a  
20 fixed position on the screen and relative to the image on the screen. It is important to note that hotspots in panoramic video productions have a fixed position relative to the panoramic frame rather than to a fixed position on the screen. Further, in true virtual reality productions the hotspot is a characteristic of the object rather than a screen or frame position.

Until the present invention, the means for moving the hotspot location in a video  
25 environment was simply not needed because the video used in multimedia productions was either dramatic video which was not navigable or was a single still frame used as a stationary bitmap. However, with the invention of navigable spatial video, a need has arisen for means to locate the hotspot on multiple frames of video and permit movement of the hotspot from frame to frame in coordination with the changing location of environmental features in the  
30 environment. Stated otherwise, in view of spatial video and continuous play video as provided by the present invention, there develops a need to scale hotspot size and track

hotspot position across a large number of frames during movement of both the viewer and the scene being viewed as well as the perspective of the viewer looking at that moving scene.

The maintenance of visual perspective is important to maintaining the user's sense of "being there." Old type stationary hotspots which are defined by the static region of the display which they occupy are referred to herein as "region hotspots" to distinguish them from the hotspots of the invention which track environmental features from frame to frame. Region hotspots have some uses even in spatial video productions created according to the invention, especially in areas of the display outside the portion of the display in which the spatial video is running.

### Receptacles

Receptacles are another way to permit user participation in an interactive multimedia production. In contrast with a hotspot, discussed above, a receptacle is a location in a video scene into which or upon which graphical overlays or graphical "objects" may be placed.

Receptacles in the prior art have a size and a position which are defined relative to absolute position and measurement in the scene for which the receptacle is defined. However, in spatial video, prior art receptacles are not adequate. A spatial video receptacle has a position, an angle of view, and a scale factor, all of which will vary from frame to frame in the spatial video. The position of a receptacle, like that of a hotspot, is recorded relative to the dimensions of each video frame in which it appears. The scale factor is a dimensionless number that will determine the ultimate size of an object rendered into the receptacle. Use of a scale factor allows objects of different absolute sizes to be placed in the same receptacle and retain their relative sizes while the actual size of the object image will increase or decrease depending on the receptacle scale factor. Further, to allow a receptacle to "hold" objects of different absolute sizes, the actual area of the display taken by the object in the receptacle will vary depending on the absolute size of the object and the receptacle scale factor and will be different for different sized objects. The angle of view is measured in degrees and allows different views (faces) of an object to be rendered into a receptacle when it is seen from different points of view in different video frames. For example, suppose a spatial clip recorded the view of moving down a corridor and passing by a table. Suppose the developer chose to render an object so that it appeared to be sitting on the table. In order

for the rendered object to appear natural, it would have to grow in size as the camera gets nearer to the table. The rendered object would also have to move in each frame just as the image of the table moves in each frame. Finally the object would have to be rendered from different angles of view, just as the table is seen from different angles of view as one passes it.

The current state of the art does not provide for these size or perspective changes in order to maintain spatial integrity from the viewer's point of view as the viewer navigates the environment.

## 10 Objects

Use of objects are yet another way to enhance user participation in an interactive multimedia environment. As mentioned above, hotspots allow the user to interact with objects in the environment. Simply by clicking the mouse on the appropriate spot, usually on the object, the user can "pick-up" and examine the object, rotate the object, and view it from different angles. Typically, prior to the present invention, when you click on one of these objects, a special window opens on the screen, the object appears separated from the navigable environment, such as in a separate window, and you can rotate the image 360 degrees horizontally, and a lesser amount vertically. The points-of-view are all dependent on what has been photographed or video taped and then linked into the scene of the environment. In the case of *QuickTime VR*, numerous photographs of the object are shot, but this rotatable object only appears in the panoramic environment as a 2-D feature of the panoramic still picture. The user uses a mouse click over the hotspot defined to cover the depiction of the object on the panorama itself. This mouse event calls a subroutine which accesses the file containing the object rotation pictures. Then a window opens containing the rotatable object separated from the environment. According to the definitions of hotspot and receptacle used in this disclosure, the area covering the environmental feature which is the subject of the rotatable image is a hotspot, not a receptacle, because no graphic is overlaid on the scene in the defined area. Instead, clicking on the hotspot leaves the area unchanged, but calls a flow routine which opens a separate window or visual in which a 360 degree rotatable depiction of the environmental object is contained.

In the case of *Surround Video*, objects can be filmed on video tape or rendered then the video containing such objects can be rendered on top of a still panorama background, using blue screen or reserved color techniques. As the object is rendered or overlaid onto the scene, the area defining the place where the object is to be placed is a receptacle rather than a hotspot. Since in the prior art a receptacle, just as a hotspot, is in a fixed position on the screen, it follows then that objects placed into the receptacle have the same limitation. Both *Surround Video* and *QuickTime VR* use panoramic video rather than rendered bitmaps. As discussed above, both of these systems allow for the use of a sequence of still panoramas to simulate movement. However, hotspots and receptacles must be defined manually in each panorama. Thus, like other prior art techniques, the panoramic still video technologies use individually defined hotspots and receptacles in fixed positions on the panoramic bitmap. Using the techniques of the prior art it was impossible to have objects (whose source media can be any media, not just photographs or video clips) placed into receptacles, and scaled and tracked across a large number of video frames during movement of both/either the perspective of the viewer and/or the scene (object) being viewed.

#### Use of Hotspots, Receptacles and Objects in Spatial Video

For illustrative purposes, assume a scenario where a motorcyclist, doing stunts with his motorcycle, is jumping off the elevated end of an inclined ramp. Further, for illustrative purposes, assume it is desired for an interactive multimedia participant viewer to be able to place a miniature guardian angel on the motorcyclist's shoulder just as the cyclist leaves the ramp. Next, assume that the angel is to remain on the cyclists shoulder as he executes the jump and safely proceeds on his way, The motorcycle, the rider, the ramp, and the surrounding environment are all part of a scene from a video tape recording. The guardian angel is a computer generated rendition and subject to placement and scaling under control of a computer program.

The viewer, standing at a distance from, but on the surface to which the rider will jump, is to observe the described action. It is an objective that the spatial perspective of the scene remain true to the viewer.

Rendering of an object, the angel, is well within the capabilities of known tools by those skilled in the art. Methods and tools for definition of a receptacle, a location within a

scene into which an object may be placed, on a per-frame or sequence-of-frame basis, is also well known to those skilled in the art. However, to have the rider on his bike, a receptacle defined on his shoulder, a guardian angel made to appear in that receptacle as the rider approaches the point of danger, all to remain in true spatial perspective as to size and viewing angle on every frame of the motion video being played as the rider approaches the viewer, executes the jump, passes the viewer, and proceeds on his way has not been possible with prior techniques and tools except by manual insertion, on a per frame basis.

### User Interface

In the context of the present invention there is first a user interface for the ultimate user, the person who will "use" the production. Use of the production could include such activities as playing of a game, "touring" of a museum, or "shopping" at an eclectic department store such as Harrods in London. A superset of this end-user interface is an interface to be used by the people that conceive and make the production. Thus, the superset user interface includes interfaces for purposes of authoring, capturing, editing, and binding the production.

The basic elements of user interfaces: input devices, interaction techniques, and interaction tasks are very well dealt with in Chapter 8, pp. 347-389 of the textbook *Computer Graphics, Principles and Practice*, by J. Foley, A. van Dam, S. Feiner and J. Hughes (Second Edition, 1990) published by Addison-Wesley Publishing Company, Inc. (herein after "Foley & van Dam"), the disclosure of which is hereby incorporated herein by reference. Still further, Foley & van Dam in Chapter 9, pp. 391-433 discusses how to assemble those input devices, and interaction techniques and tasks into a usable and pleasing form for a user interface.

However, the design and generation of an interface and the use of that interface wherein there is developed a direct correlation between a location within a topography and the selection of a video clip and a frame from within that clip which was shot with the camera at said location has not heretofore been anticipated. This, then, provides yet another opportunity for invention. As will be seen later in the description of the invention, this opportunity is answered in the authoring and editing steps of the present invention wherein such an interface is presented in the form of a map-based viewer and a map-based editor.

### Camera Tracking

The process of creating spatial video according to the present invention can be enhanced by recording the coordinates of the camera used to shoot the film or video to be included in the spatial video environment and associating these coordinates with the video frames shot. Of course, where rendered scenes are being captured, virtual camera coordinates for each frame can be generated by the rendering program.

Camera tracking is the detection and recording of the position and orientation of the camera in the environment where film or video is being shot (captured). There are a variety of means known in the art for tracking the position of an object in three dimensional space. Most of these means can be adapted to locate a camera moving in three dimensional space and record its location and attitude over time. The resulting positional data can then be associated with the frames shot to provide a camera position and a camera attitude measurement for each frame. The same or similar techniques can be used to determine the position of relevant features in the environment being captured with which hotspots, receptacles and the like are to be associated.

One approach that could be used to provide at least a partial solution to the camera position and angle problem would be computer controlled camera mounts. These provide exact positional data but tend to be immobile, cumbersome and expensive. However, if the capturing is being done on a sound stage or other production facility equipped with such equipment, it certainly can be used to accomplish this feature of the specification. Use of such a camera mount to generate positional data is obvious. However the camera mount can also be used to generate positional data for features. This can be accomplished by making closeup shots of environmental features as the camera traverses the dimensions of the feature. The resulting position data can then be transferred to a feature position file for use in the production.

Where capture is to occur in a place not equipped with computer controlled camera mounts, less expensive and more flexible tracking means are available. These other means for determining position fall into two major categories: relative and absolute position measurements. Relative position measurement methods include dead-reckoning and inertial

navigation. Absolute methods include active beacons, artificial landmark recognition, natural landmark recognition, and model matching or map-based methods.

Dead Reckoning. It is well known that dead-reckoning provides good short-term accuracy, is inexpensive, and allows very high sampling rates. However, the fundamental  
5 idea of dead-reckoning is the integration of incremental motion information over time, which leads inevitably to the accumulation of errors. Particularly, the accumulation of orientation errors will cause large position errors which increase proportionally with the distance traveled by the object being tracked.

Despite the accumulation of errors inherent in dead reckoning methods, they could, in  
10 certain situations, provide sufficient accuracy to be used for camera tracking purposes in the present invention, but they are not considered a preferred embodiment.

Inertial Navigation. Inertial navigation systems, also relying on integration of rate-data to yield position, similarly result in a small constant error increasing without bound over time. Efforts to minimize error sources result in high manufacturing and maintenance costs  
15 for these systems. However, in recent years, the prices for very accurate laser gyros and optical fiber gyros have come down significantly and the devices may be suitable for camera tracking applications.

Active Beacon Navigation Systems. Active beacon navigation systems can be highly reliable, provide very accurate positioning information with minimal processing, and allow  
20 high sampling rates. Active beacon systems include radio frequency, laser and ultrasonic systems. One radio frequency based system which can be used to measure camera location is the Navstar Global Positioning System (GPS) which uses a constellation of 24 satellites orbiting the earth every 12 hours at a height of about 10,900 nautical miles. Error sources in the GPS include satellite position, ionospheric refraction, tropospheric refraction, multipath  
25 reflection and selective availability. All of these error sources, with the exception of multipath effects, can be virtually eliminated through the use of differential GPS (DGPS). Various types of DGPS achieve various degrees of accuracy, but among them, the codeless phase differential reference using the phase of the two carrier frequencies achieves accuracy of about 1 cm.



GPS is excellent for outdoor navigation, but is questionable for use in areas where RF reception difficulties may interfere with reception of its signals, such as indoor use, dense foliage or steep or mountainous terrain.

Local active beacon systems generally require modification of the environment in order to put the required stationary sensors (or transmitters if the active beacons are fixed). Further, line-of-sight between transmitter and sensor needs to be maintained between the beacon on the vehicle and two or more of the sensors.

There are several commercial active beacon systems available using laser, infrared, and ultrasonic transducers. Under appropriate conditions many of these commercial systems can provide high accuracy and high reliability in vehicle tracking.

One can distinguish between two different types of active beacon navigation systems: trilateration and triangulation. Triangulation generally uses two or more active beacons at fixed locations a known distance apart. The moving object contains a directional receiver with which the bearings to the two beacons can be measured. This gives the values of one side and all angles of a triangle from which the position can be computed. Trilateration is the determination of a moving object's position based on measuring the time-of-flight of a pulse of energy traveling from a transmitter mounted on the vehicle to multiple receivers mounted at known locations in the environment. Conversely, there may be multiple transmitters mounted at known locations in the environment and one receiver mounted on the vehicle. Also, a transmitter and a receiver may be co-located on the vehicle and the time measured is a round-trip time-of-flight of the energy bouncing off multiple reflecting objects mounted at known locations in the environment.

Time-of-flight systems include both laser-based and ultrasonic systems. Using time-of-flight information, such systems compute the distance between the moveable transducer on the vehicle and the stationary transducers mounted in the environment by measuring the time of flight of the light or sound wave. The relevant parameters involved in range calculation are, for ultrasonics, the speed of sound in air, and for lasers and radio frequency system, the speed of light. The speed of sound is markedly influenced by temperature changes, and to a lesser extent, by humidity. For example, an ambient temperature shift of just 30 degrees F. can cause a 0.3 meter error at a measured distance of 10 meters. Where

ultrasound-based systems are used, significant temperature changes can be caused by use of movie lights during filming.

Ultrasonic Trilateration. As can be seen, there are several methods familiar to those skilled in the art which can be used to track the camera and record its position. As will become clear as the invention is discussed, this data is useful for subsequent correlation with video (or other picture) frames and environmental features in order to accomplish the purpose of the present invention. Where the environment to be captured is relatively small and free from significant obstructions to interfere with wave propagation, ultrasonic trilateration is the preferred embodiment.

An ultrasonic transmitter can be mounted on the camera, for camera tracking, or positioning wand, for recording positional data on relevant features of the environment. Receivers can be mounted in locations to minimize obstructions between the camera and sensors. Exact positioning of sensors will vary from environment to environment, however, where the environment being recorded is a room good results can often be achieved by mounting the sensors on the ceiling.

Prior art algorithms for use in ultrasonic trilateration employ a least squares estimate and are valid only if the expected values of the errors in the measurements is zero. When tracking the ultrasonic transmitter over a large area containing line-of-sight obstructions, and where the vehicle while traversing a camera path is moving into and out of range of one or more sensors, the positional data generated using prior art least squares algorithms contained sufficient false signals and large errors to seriously affect its viability in a camera tracking system. One aspect of the current invention is a more robust method of determining location under such circumstances.

## Multimedia Process Overview

Today's multi-media productions, such as interactive computer games, are typically created in three major steps: determination of content, production of content and assembly of production. The writer, art director, and the producer create all the pieces (film, art, sound), then pass them on to the programming team, which assembles the product for a specific system, such as Microsoft Windows or the Sega Saturn. If the final product does not sell

well on the target system, the pieces must then be reassembled for another target system, often by different programmers and at great expense.

There is a need in interactive multimedia systems for a single integrated system for writing, storyboarding, capturing assets, editing, and mastering a multi-media production and for enhancing the user's experience as the production is played, and which can be implemented using low cost personal computers. The present invention provides such an integrated process.

### SUMMARY OF THE INVENTION

It is therefore an object of this invention to enhance an interactive multimedia user's perception of being in a presented environment by providing a computer-readable medium for causing a computer to generate an interactive-multimedia world displaying a high resolution video-based environment capable of being spontaneously navigated in a continuous fashion in response to user input without being limited to making transitions between predefined panoramas or otherwise using only transitions which have also been previously recorded by the producer.

It is another object of this invention to provide an interactive world using motion video to create the interactive environment of the world.

It is a further object of this invention to provide a method for enhancing the navigational realism of the motion video environment by providing smooth transitions between various video clips and frames of the environment.

It is another object of this invention to provide for capture of motion video using cylindrical projection by lens or computer rendering technology to produce video frames which can be joined side by side without leaving a fracture at the joint.

It is another object of this invention to provide an interactive motion video environment wherein groups of motion video clips are generated by moving a camera (or virtual camera) along a path through the environment, each group of clips providing a multiplicity of viewing directions for the path.

It is another object of this invention to provide an interactive motion video environment containing paths represented by a clip group comprising a plurality of video clips representing a plurality of camera headings of the path through the environment

wherein the user has the option to transition from clip to clip within the group creating the impression of a seamless rotation of view on a single path in the environment.

It is another object of this invention to provide an interactive motion video environment comprising a plurality of clip groups representing a plurality of paths through the environment wherein the user has the option to transition from a clip in one clip group to a clip in a second clip group at a point of intersection of the paths which said clip groups represent providing the navigating user with the visual impression of a seamless turn from one path in the environment to another.

It is another object of this invention to provide a spatially-oriented process for the authoring, capturing, editing, compiling/binding, and running or playing of high-resolution video-based interactive multimedia productions using a multiplicity of means for these activities; said multiplicity of means to include topography, and to provide productions created according to said process.

It is a further objective of this invention to provide a process whereby a topography-based means for authoring, capturing, editing, compiling, binding, and running or playing of interactive multimedia production can be used in conjunction with heretofore known time and/or logic-based means for these activities.

It is another objective of this invention to provide means for defining clips and clip groups on a map of the spatial video environment, for assigning video clips to the various map-defined clips and clip groups in the environment and for defining spatial relationships between the clips.

It is an objective of this invention to provide camera tracking means to determine and record position of the camera in the environment where film or video is being shot (captured), said means also including means for correlation of the position data with each frame of film or video.

It is a further object of this invention to provide a method for creating, editing and displaying hotspots which appear, move, and adjust in size in order to track the position and size of the video feature with which they are associated from frame to frame within a video environment.

It is a further object of this invention to provide a method for creating, editing and displaying receptacles which appear, move, and adjust in size and orientation in order to

track the position and size of the video feature with which they are associated from frame to frame within a video environment.

It is a further object of this invention to provide a method for creating, editing and displaying objects which appear in true spatial perspective as to size and viewing angle for the point of view of the user from frame to frame within a video environment.

These and other objects are provided in a system for creating an interactive multimedia production, said production including an environment, displayed in high resolution video, which a user can freely explore and productions created using such a system.

### BRIEF DESCRIPTION OF THE DRAWINGS

Other objects, features and advantages of the present invention will become more fully apparent from the following detailed description of preferred embodiments, the appended claims and the accompanying tables and drawings in which:

**Table 1** is a table showing an example of disassembled code from the logic hunk of a production made according to a preferred embodiment of the invention with instructions loading the user interface of the production;

**Table 2** is a table showing an example of disassembled code from the logic hunk of a production made according to a preferred embodiment of the invention showing instructions running parallel with spatial video clip playing, illustrating certain types of interesting frames including hotspot activation instructions;

**Table 3** is a table showing an example of disassembled code from the logic hunk of a production made according to a preferred embodiment of the invention showing instructions running parallel with spatial video clip playing, illustrating other types of interesting frames including cache frame instructions;

**Table 4a** is a table showing the flow language instructions used to make a simple flow routine according to a preferred embodiment of the present invention;

**Table 4b** is a table showing the actual code a binder according to a preferred embodiment of the present invention generated from the flow language instruction of **Table 4a**;

**Table 5** is a table showing a disassembly of part of the hotspot data table from the Data hunk of a production made with a preferred embodiment of the present invention;

**ATTACHMENT 1** contains illustrations of the appearance of the various menus used in the preferred embodiment of the edit system;

5       **FIG. 1** is a block diagram of a multimedia computer system for editing and displaying spatial video productions according to the present invention;

**FIG. 2** is a block diagram of the generalized steps of the process of creating and running a spatial video production according to the present invention;

10       **FIG. 3a** is a side by side depiction of two pictures of adjacent areas of a room rendered using a standard rectangular projection 90 degree field of view representing a total horizontal field of view of 180 degrees;

**FIG. 3b** is a portion of a cylindrical projection according to the invention showing the same 180 degree horizontal field of view of **FIG. 3a**;

15       **FIG. 4a** shows the uncorrected distortion of vertical lines in the cylindrical projection image of a grid;

**FIG. 4b** shows an ideal cylindrical projection image of a grid;

**FIG. 5** is a block diagram of the camera tracking apparatus and camera tracking control computer of a preferred embodiment of the present invention;

20       **FIG. 6** is a block diagram showing the capture and camera position data correlation apparatus and process of the present invention where a motion picture film camera is used in the initial capture;

**FIG. 7** is a block diagram showing the capture and camera position data correlation apparatus and process of the present invention where a video camera is used in the initial capture;

25       **FIG. 8** is a process flow diagram showing the iterative push-pull positioning process of the present invention;

**FIG. 9** is a process flow diagram showing the steps in the creation of an M-view according to the invention;

30       **FIG. 10** is an illustration of the catalog directory structure used to store M-views and visuals according to a preferred embodiment of the present invention;

**FIG. 11** is an illustration of the inputs and outputs of the spatial video edit process according to a preferred embodiment of the invention;

**FIG. 12** is a general process flow diagram of the edit process according to a preferred embodiment of the invention;

5       **FIG. 13** is an example of a spatial video map of an environment in a spatial video production as it might appear in the map editor or viewer according to a preferred embodiment of the present invention;

**FIG. 14** is a process flow diagram showing the steps for laying out clips on a map according to a preferred embodiment of the process of the present invention;

10       **FIG. 15** is a process flow diagram showing the steps for associating a video clip with a map according to a preferred embodiment of the present invention;

**FIG. 16** is an illustration depicting possible transitions from various clips with a clip group and between two intersecting clip groups according to the invention;

15       **FIG. 17** is a process flow diagram showing the steps for detecting clip intersections according to a preferred embodiment of the invention;

**FIG. 18** is a process flow diagram of the runtime process in a production made according to a preferred embodiment of the present invention;

**FIG. 19** is a process flow diagram of the runtime flow interpreter of a preferred embodiment of the present invention;

20       **FIG. 20** is a process flow diagram of the runtime video interpreter of a preferred embodiment of the present invention;

**FIG. 21** is a process flow diagram of the DOFRAMES routine of the runtime video interpreter of a preferred embodiment of the present invention;

25       **FIG. 22** is a process flow diagram of the DOEVENTS routine of the runtime video interpreter of a preferred embodiment of the present invention;

**FIG. 23** is a process flow diagram of the runtime asynchronous events handler process of a preferred embodiment of the present invention;

**FIG. 24** is a process flow diagram of the runtime visual load routine of a preferred embodiment of the present invention;

30       **FIG. 25** is a relationship chart showing the relationships of the routines and processes of **FIGS 18 - 24**;

**FIG. 26** is a block diagram showing the various types of assets to be identified and brought in to the spatial video edit process;

**FIG. 27** is an illustration of a display screen showing the 7 navigational regions of a preferred embodiment of the invention;

5 **FIG. 28** is a process flow diagram of the overall binder process of a preferred embodiment of the present invention;

**FIG. 29** is a process flow diagram of the data hunk creation steps of the binder process of a preferred embodiment of the present invention; and

10 **FIG. 30** is a process flow diagram of the logic hunk creation steps of the binder process of a preferred embodiment of the present invention.

### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The process of the invention is based primarily on the inventive concept of navigable motion video. A navigable motion video production is a motion video-based representation  
15 of an environment in which the user of the production is provided with a degree of freedom of movement or freedom of viewing direction within the environment not previously available in motion video. The freedom of movement or viewing direction within a navigable motion video environment is characterized by the availability to the user of one or more navigation abilities not present in prior art motion video. Such abilities include,  
20 without limitation, the ability to travel along a path within the environment in either direction; the ability to pan left, or right beyond the initially displayed field of view while travelling along the path; the ability to pan up or down while travelling along the path; the ability to turn onto an intersecting path; the ability to change direction of travel along a path; the ability to stop at a point along the path and to rotate viewing up to all 360 degrees of the  
25 environment in the plane of rotation; the ability to interact with hotspots in the environment which maintain their proper associations with environmental features regardless of where on the display screen and what size the feature is at the particular location along the path and particular portion of the field of view being displayed; and the ability to interact with objects inserted in the video environment which objects maintain a proper size, angle of view and  
30 location within the environment regardless of the location along the path and the particular portion of the field of view being displayed.



Prior to the invention "navigation" of motion video was confined to merely starting and stopping the play of a motion video clip, playing the clip in reverse, and playing the video clip in either direction at a faster or slower speed than normal. These methods of playing video are not included within the concept of navigable motion video.

5       The process of the invention is based further on the inventive concept of spatial video. Spatial video is a preferred embodiment of navigable motion video which is a way of generating and playing motion video to give the impression of being immersed in a video world in which one can move around at will.

10       The scenes presented in spatial video may show actual places, movie sets, or places that never existed but were modeled in a computer. The user's ability to move around at will includes the ability to stop or move ahead at user-controlled (or user-requested and computer-selected) speeds, the ability to look left and right, the ability to turn around, and the ability to turn left or right and follow a new path. When turning onto a new path, the video will show a smooth turning from the view looking along the first path to the view  
15       looking along the second, just as a person would see when making such a turn in the real world. Every video frame that the user sees will follow naturally from the preceding frame (no sudden jumps in the video) and will contain no discontinuities: no seam lines introduced in the video; no fractures where outlines of objects take unnatural bends.

20       A spatial video clip is captured by moving a camera (or virtual camera) along a path through a world or environment. Typically a lattice of intersecting paths will be captured in this way. During playback, the user will be limited to moving only along the lattice of video paths. At any intersection he will have the option of proceeding on the current path or turning right or left onto the intersecting path. If the lattice is dense enough, the viewer will have the ability of moving to virtually any point in the scene. The paths in this lattice are  
25       preferably placed on the natural navigational paths of the world or environment.

30       The spatial video of the invention differs from normal linear video in several key aspects. Unlike linear video, spatial video is navigable and can include overlaid objects which retain their position and orientation relative to the features of the environment recorded on the video frames and the player's point of view. Unlike panoramic video, spatial video consists of motion video rather than photographic still shots. In the preferred

embodiment, the navigable feature is coupled with the concept of capturing spatial video along the natural paths of the environment as discussed more fully below.

The concept of natural paths is important to the preferred embodiment of the invention. The principle is to capture spatial video of the environment from the perspective of a person navigating through the environment. As a person navigating the environment will not view the environment from every conceivable point within the environment, spatial video does not need to be captured from the perspective of every single point in the environment.

Where the environment is one suitable for walking, the environment is preferably captured on film or video from the perspective of a person walking through the environment.

For example, if the environment is a room with a large table in its center and a door on either end, the natural paths through the room might include a path from one door to the other around the table to the left and a path between the same two doors around the table to the right. While it would be technically possible to capture the environment from the point of view of a navigational path through the middle of the table. However this would not be a natural path and use of such captured footage in a production would not only unnecessarily consume storage resources but would also detract from the sense of realism of the navigation. Further, the environment is preferably captured from the point of view of a person of ordinary height walking along the path. It would also be possible to capture the environment from the perspective of a 10 foot tall person or a 1 foot tall person. However use of such footage in the production would not contribute to the realism of the navigation of the environment and would consume storage resources.

The concept of natural paths extends beyond rooms in a house to virtually every environment that a human confronts. Further examples include a formal garden environment which usually has walkways which would be the navigational paths through the environment. Wild environments, such as forests or jungles, also lend themselves well to natural path navigation. Most human navigation in such environments is along forest or jungle paths or, at least, through the natural breaks in the vegetation. Consequently, such environments can be captured on spatial in a manner which allows realistic, thorough exploration without capturing every feature of the environment from every possible perspective.

One feature of natural path through the environment is that it is capable of being traversed in two directions. One can walk from door A to door B, or one can walk from door B to door A along the same path. Another feature of a natural path is that the person traversing the path may be looking ahead or to the side, or may turn completely around.

5 Thus video information is preferably captured providing a panoramic field of view from every point along the path. "Field of view" as used herein is not meant to be limited to the view captured by a single camera at a single time, such as the field of view captured by a wide angle lens, but is meant to encompass the total field of view captured from a single point within an environment regardless of the number of cameras used, the field of view of  
10 an individual camera lens or lenses used, or the number of camera headings or positions used to capture the environment from that point.

Another feature of natural paths is that they often intersect. Thus, in more complex environments there may be numerous places where paths cross. Thus a top down map of the environment on which the natural paths are represented as lines may have numerous  
15 intersections and, consequently, may appear as a lattice of paths. An important feature of a spatial video environment using natural paths is to allow the navigator to change paths at any such intersection. Another important feature is to allow the navigator to not only turn onto any intersecting path, but also to turn around and retrace his/her steps at any point along a path.

20 According to the process of the invention, spatial video is created by first capturing the environment using the capturing system of the invention. While the system has great advantages in efficiency and cost savings when used to capture physically existing environments, it also provides advantages when some or all of the environment being captured is comprised of rendered scenes. Rendered scenes or portions of scenes can be  
25 easily integrated into a spatial video clip of the present invention. Physically existing scenes such as those shot on location or those created on sound stages can be captured on film or directly on video. In the preferred embodiment, as discussed more fully below, such scenes are captured on film using a motion picture camera running at 24 frames per second and equipped with a cylindrical projection lens.

30 In the preferred embodiment spatial video clips are captured by moving the actual or virtual camera through the environment to be captured along the natural navigation paths of

that environment. The benefit of using natural paths is that a significant reduction in the number of paths to be captured can be accomplished without the player experiencing any appreciable reduction in navigational freedom. This allows for a rich environment to be captured at a significant reduction in the video resource requirements.

5           The natural paths must be captured on video clips representing a sufficient diversity of camera headings to capture all of the visual information which a person might view when negotiating the natural paths in different directions and looking in different directions. Less usual forms of navigation may or may not be allowed in a particular production. For example, capture of a natural path from point A to point B with the camera heading toward point A allows the user to "walk" from A to B looking toward B. Capture of the same path from B to A with the camera heading toward A allows the user to walk from B to A facing A. Capture left and right "side facing" views allows the user to face left or right at any point along the path. In the preferred embodiment discussed herein capture of these four headings along the path provide 360 degrees of visual information at every point along the path.

10           However, unless the reverse MPEG process of the co-pending Special Effects Applications is used, this capture does not permit the user to travel from A to B facing toward A, that is backing up rather than walking forward. There may be circumstances in a particular production where such travel should be allowed, for example, where the program is simulating driving a tank driving in reverse would be desirable. Thus, to adequately capture a particular environment, not only must the natural paths be determined, but also the nature of navigation to be provided in the particular circumstances.

15           Use of a wide angle lens, with appropriate optical corrections before playback allows for the capture of sufficient visual information on a single frame to allow the user some panning navigation, similar to that described above in the discussion of panoramic video. As mentioned above, use of a cylindrical projection lens, or rendering system where images are computer generated, is preferred. This type of projection produces images which, if taken from the same node location can be joined side-by-side without leaving a fracture at the joint. This allows for the smooth transitions between adjacent video clips which is an important feature of the invention.

20           There are two primary categories of smooth transitions in spatial video productions. One is the turn transition, where the player makes a transition from one frame of a video clip

shot on one path (the FROM frame) to a frame on another video clip shot by a camera (or virtual camera) traversing an intersecting path (the TO frame). The turn transition has the visual effect of a turn from one natural path to another. The second is the roundabout, where the player makes a transition from a FROM frame of a video clip shot on a path by a camera heading, for example, north, to a TO frame on another video clip of the same path shot by a camera heading, for example, east. The roundabout has the effect of a rotation where the player turns in place on a single path. Roundabouts may be repeated, for example, from east to south, followed from south to west and back to north, creating a 360 degree rotation. The mechanics of such transitions are discussed in more detail in the following sections of this disclosure.

Also central to the creation of spatial video is the need to precisely define the paths the camera (or virtual camera) traverses in capturing the environment so that the node of the camera remains in the same position relative to the path for all of the video clips captured on the path, regardless of the heading of the camera. When coupled with the preferred use of cylindrical projection, this allows for production of seamless transitions between video clips for "FROM" frames and "TO" frames shot (or rendered) from the same camera node location. If the camera node varies, the smoothness of the transition suffers.

Also central to the creation of a seamless spatial video environment is enablement of "transitions-on-the-fly." When the viewer tells the system to "turn right" onto an intersecting path, spatial video shows a smoothly turning angle-of-view that brings the viewer's viewpoint into alignment with the target path views. For example, when a viewer going north wishes to turn right (east), the system shows a succession of frames that all represent views from the viewer's current position, but looking in different directions. These directions start with north (zero degrees) and end with east (90 degrees). The term, "transitions-on-the-fly" refers to a technique for generating this sequence of frames - in a format suitable for the playback engine - using only the beginning frame and end frame as inputs. All the intermediate frames in the sequence are combinations of the first and the last frame created according to the invention disclosed in the Special Effects Applications. Further, by caching TO frames, the transitions on the fly can be accomplished before or while the system is seeking and loading the TO video clip. Because the transition can be accomplished without use of video resources other than the FROM frame which is stored in

a streamer or MPEG player buffer, the TO frame which is cached and then read into the buffer, and a short, preferably previously created generic transition sequence which is the same for all transitions of a particular type (such as left turn), the transition sequence can be used to mask seek time, loading time, clip set up time and the like with a meaningful transition.

An additional use of the transition is to overlap the loading into memory (see runtime description below) of the cached TO frames of the destination clip with the transition. In a preferred embodiment of a production according to the invention the frames representing the TO frames for all allowed transitions off of the clip being accessed are placed at the front of the clip. These are designated as cached frames and the MPEG frames are loaded into the cache memory before the starting frame of the clip is decoded and displayed. These loading tasks can be interleaved with the transition sequences being sent to the decoder. Thus, transition sequences can be used to mask delays caused by several different types of runtime operations.

The most difficult aspect of the transition technique is ensuring that the generated frames use a format suitable for the video playing technology. If an MPEG player is used, for example, the system must generate a sequence of valid MPEG frames containing the required views. This aspect of the technique, for all digital encoding formats using reference frames and dependent frames is the subject of the Special Effects Applications incorporated herein by reference. Another important aspect to this technique is the caching of "TO" frames. In the preferred embodiment the possible "TO" frames which could be the subject of the clip currently being viewed are cached in memory as the clip is loaded. As a result, transitions can be immediately executed upon receipt of the instruction without the prior need to seek and load the "TO" clip. As a consequence, during the time that the transition frames are being played, the system can prepare for playback of the frames in the new video path. This preparation may involve file seeking, buffer loading, video parsing, and the like.

While excellent spatial video productions can be produced using only the path capture, cylindrical projection and transition technology described above, the present invention also includes enhancing technologies which allow for the creation of spatial video productions more efficiently and with more realism. The first such enhancement is the use

of camera position data. If exact locations of the camera are known for every video frame, then automation may be used in locating intersection frames and in matching frames from different clips in a group of clips taken along a single path to generate smoother transitions and to generate transitions more quickly.

5           A second enhancement is the use of wide angle lens projection. If each frame capture 90 degrees of view, then the entire scene (panorama) may be captured in four frames. If two video paths intersect at right angles, and if each path is captured in both directions, using a 90 degree field of view lens, there is sufficient video for a full panorama (complete 360 degree turn-around) at the path intersection. Moreover if each path is captured four times  
10 (looking ahead, looking behind, looking right, looking left) with such a lens, then panoramas will be possible at any point along any path. These panoramas allow the viewer to turn and face any direction during playback, and to turn around and proceed along the path in the opposition direction.

          Capturing a wide field of view (particularly 90 degrees) is useful for turnarounds and  
15 intersections as explained above. If the lens also captures an equivalent angle of height, and if traditional video aspect ratios (four by three) are being using then the lens might also be expected to capture about 67 degrees of height. (67 is : of 90). This is a large angle of height to capture. Such a view would usually include a lot of ceiling at the top and floor at the bottom, with the "interesting" parts of the scene compacted into the middle. A better  
20 solution is an anamorphic lens that will capture 90 degrees of width but a "normal" viewing angle for height (say 34 degrees). This would capture all the width we need without capturing excess (and uninteresting) height. Of course the resulting anamorphic distortion would have to be removed at playback time by stretching the video horizontally.

          The use of a wide field of view also enables another enhancement to the spatial video  
25 technology. This is the ability using panning techniques to allow the player to navigate to the left and right within the video frame. Panning within the frame coupled with use of cached TO frames and the transition techniques of the Special Effects Applications allow for a remarkable degree of navigational realism.

          Another enhancement is through the use of interpolation and extrapolation techniques  
30 to place hotspots, receptacles and other types of overlays in the proper location with the proper size on each frame of a clip. As the camera approaches the object with which a

hotspot or receptacle is to be associated, not only the position of the object on the screen changes but also the size of the object. The techniques of the invention allows precise hotspot and receptacle location and sizing without having to draw manually draw the hotspot or receptacle on every frame of video.

5 A further enhancement is through the use of angle interpolation and the M-view structure, more fully described below, to have sprites inserted into the video maintain their proper size relative to the background on which they are superimposed as well as the appropriate face. Thus, as the view approaches the object in the video environment with which an M-view sprite is associated, the profile of the object visible to the player changes  
10 with the player's location in the spatial video environment. For example, assume the sprite is a statue. As the player approaches the statute the right profile may be visible. As the viewer passes the statue the front profile will be visible. At intermediate points, faces of the statue between the right profile and front view will be visible.

## 15 System Environment

The Computer. The present invention is designed to permit the efficient and cost effective development of spatial video productions and to permit the user to navigate freely within that environment along the natural paths contained therein, using a low-cost personal computer. Although the present invention may also be used, after porting to the appropriate  
20 software environment, to create and run spatial video productions on game consoles, high-speed workstations, mainframe computers or supercomputers, it was designed to provide this capability in the personal computer environment. Particularly, the preferred embodiment was designed for operation in the Microsoft Windows 95™ operating environment and for use on computers compatible therewith.

25 The invention is preferably used on any personal computer having a CPU clock speed of at least 90MHz, at least 8 Mbytes of RAM, at least 50 Mbytes of available hard disk space, and a CD-ROM running at a minimum of 2X speed. The computer also preferably contains a hardware decoder either on its motherboard or as an expansion board. Referring now to **FIG. 1**, there is shown a block diagram of a video system comprised of a multimedia  
30 computer system **10** which could be employed to implement the present invention. In **FIG.1**, a computer system **10** is shown having a Pentium7 100Mhz CPU based computer **1**



with an 850MB internal hard drive, a 4X CD ROM drive 8, a disk drive 9, a SVGA monitor 2, speakers 3, a keyboard 6, and a pointing device such as a mouse 7 with two selection keys 7a and 7b which may be used in conjunction with the mouse to allow a user to navigate through the spatial video environment and to interact with receptacles, hotspots and objects in that environment. The computer could be connected to a local area network and/or modem for accessing resources not located within the computer's local drives. Implementation of other user interface devices, add-ons, operating systems and peripherals would be obvious to those skilled in the art and will not be discussed further herein.

The Video Decoder. In the preferred embodiment the computer system also incorporates a software MPEG decoder, such as SoftMotion from SAS Institute Inc. of Cary, North Carolina. A spatial video production imposes a challenging set of requirements on any video playback system used, whether software-based or hardware-based. Here are the key ones:

1. Support for highly compressed video (to minimize the number of CDs shipped)
2. High clarity/reproduction of detail
3. 24 frames per second, or higher (to avoid flicker)
4. Ability to quickly seek to a different video frame (when the user turns onto a different path, for example)
5. Ability to stretch horizontally (to compensate for anamorphic capture)
6. Ability to report exactly what frame is on the screen at any time (to allow positioning of hotspots and M-views)
7. Ability to paint M-views on top of video frames without flicker (painting the M-view directly on the displayed frame tends to produce a flickering M-view, thus this requirement is preferably solved by use of a decoder which paints to a display buffer).
8. Video sound track must be played in sync with the frames (lip sync)

The ability to accomplish these requirements in the preferred personal computer environment led to the choice of MPEG-1 as the preferred compression technology. MPEG-1 was the only compression technology available at the time the choice was made which could compress an hour of high-quality video onto a single CD. Since the time this

development decision was made, other compression technologies have been developed. The processes of the present invention work equally well with most of these other technologies.

At the time the spatial video production system was developed, there were two broad classes of MPEG player available; hardware and software. Hardware players use special video processing chips, either on the computer's motherboard, or more commonly, on a special video card in the computer. Typically these chips speed up the process of decoding the MPEG video and/or the process of drawing the decoded video on the screen. Software players, on the other hand, do most of the work with program logic, and use only conventional computer hardware to display the decoded video.

When the spatial video system was developed, neither class of video player met all the above requirements with the same proficiency. Hardware players excelled at high frame rates, video quality and sound sync, but only occasionally did stretching (with panning), seldom did flicker free overlays (M-views), never supported seeking to a specified frame, and never reported exact frame numbers. Software players usually support stretching with panning and are usually are able to report which video frame is being displayed at the moment. However, software players are not as proficient at playing video at high frame rates with synchronized audio. With a fast Pentium™ processor a software player, such as SoftMotion, will play audio-synchronized video with satisfactory results but not with the proficiency of a hardware player. Software players other than SoftMotion can be customized to support frame accurate seeking M-view overlays without flicker, though commercially available ones seldom support these capabilities.

In the absence of an optimum, single MPEG player solution, productions made with a preferred embodiment of the present invention are capable of achieving optimum results in systems which have a hardware player by using both a software player and the hardware player. One player (the software player or spatial video player) is optimized for sequences of spatial video, and the other (the hardware player) for "dramatic video". The runtime system switches between the two players to match the characteristics of the video currently being played. The spatial video player is chosen to excel in stretching, panning, knowledge of exact frames, and flicker-free M-view overlays (things a software player is good at). It is used to show free roaming sequences like moving down a corridor.

When dramatic clips are played, the runtime system can switch to a hardware player that is better at showing high frame rates with synchronized audio. In the preferred embodiment the switch is accomplished with the same flow routine which launches the dramatic video. Dramatic clips usually show brief scenes in which user interaction is disabled or restricted. They lack hotspots and M-views, and therefore allow a hardware player to be used. As software players get better and PC processors get faster, we expect to be able to achieve audio-synchronized dramatic video playback comparable to that available with a hardware player using a software player. This would allow a single software player to be used for all video, and eliminate the need for two players.

The inability of most MPEG players to perform frame-accurate seeking in MPEG streams is handled through use of the streamer component of either the system or the SoftMotion decoder available from SAS Institute Inc., Cary, North Carolina. The system can present a typical player with a single continuous stream of MPEG to play. As far as the MPEG player is concerned, no seeking is ever requested. The streamer component of the SoftMotion decoder or of the spatial video runtime engine manufactures this continuous MPEG stream by assembling MPEG data from selected segments of the MPEG clips, and from transition data. In effect, the streamer does the seeking required outside of the MPEG player. When a seek is done, the streamer joins the new MPEG frames to the old ones so that the MPEG player sees what looks like a single MPEG stream with no interruptions. The streamer also injects the "synthetic" MPEG frames generated during transitions into its output stream.

The MPEG streamer function is an important component of the preferred embodiment of the invention. Its relationship to the MPEG Player is discussed in the co-pending Special Effects Applications which are incorporated herein by reference. It essentially constructs a virtual MPEG stream out of the various source MPEG streams, including the various video clips of the production, the dramatic video and the transition sequences to send to the decoder. As a result of the use of the virtual stream the MPEG decoder never "sees" a clip end. Consequently, the MPEG decoder never needs to initialize a new clip as it otherwise would have to do every time the system changed from one clip to another. Decoder initialization sequences may cause the skipping of some frames of the new

clip and will cause a delay as the player decodes initial frames before beginning to display video from the new clip.

The MPEG streamer has the ability to search or seek clips or other video sources, duplicate, omit, or reorder pictures present in the base video stream, and the ability to insert pictures into the video stream from other sources. The functionality of the MPEG streamer used in the preferred embodiment of the invention is shown in the annotated source code for the interface definition of such a streamer contained in Appendix C and incorporated herein by reference. Further information regarding the capabilities of the MPEG streamer is contained in the Special Effects Applications and in the section below on transitions.

### The Spatial Video Production Process

Certain of the steps in process of creation of spatial video productions require additional equipment which shall be identified and discussed as necessary below.

The process of the present invention comprises five major steps. These steps have well defined components and well defined interactions between the components. We will present an overview of the five major steps including a description of the components of each step and how the components interact in the process of generating and playing an interactive multimedia production. It is useful to realize that when we say components we are referring to either hardware or software, or both, to accomplish a function. In all cases it will be obvious to one skilled in the art when the component is necessarily hardware or necessarily software or can be either hardware or software or a combination thereof.

The five major steps of the spatial video production process 11 of the invention, as illustrated in the general process diagram of **FIG. 2** are called: Concept 12, Capture 13, Edit 14, Bind 15, and Run 16. As demonstrated by the generalized process diagram of **FIG. 2** and the text below the steps of the process are not rigidly sequential. For example, while the script is initially generated in Concept 12, it may continue to be refined throughout production as editing and capture reveal problems and opportunities. Capture requests are generated by Edit 14 as well as Concept 12. Similarly, test runs may reveal further editing or capture steps which must be performed. Thus, the linear treatment here of the five main steps of the production is for convenience and does not suggest that they follow each other in rigid sequence.

### Concept Step

Concept 12 is mostly a planning step where the original idea for the production is thought out. This step covers many of the traditional film/video production tasks 17 associated with any traditional movie or video segment or computer production, such as an educational program or game, including: writing, set, scenery, dialog, costume, and lighting design. Depending on the type of multimedia production being created using the processes and devices of the invention, the script may call for the use of many conventional elements, including musical scores, dramatic video clips, static bitmaps, such as pictures or text, and other similar features common in the art.

The concept stage may require use of some of these conventional elements in a unique way in that their incorporation into the production may be through the use of some of the processes of the invention disclosed herein. For example, a conventional dramatic video clip may be played because the user has clicked on a spatial video hotspot or picked up an object contained in a spatial video receptacle. Further, an animated video or dramatic video may be associated with a hotspot or contained in a receptacle.

Thus, the concept stage results in a spatial, map-oriented script 18 including many conventional elements related to each other in ways which are often unique to the spatial video technology disclosed herein. The concept step also includes the substep of laying out the spatial scenes, determining the spatial, temporal and logical characteristics of various components of and objects in the spatial scenes and spatially, temporally and logically relating the scenes to each other.

The spatial video script 18 is drawn in or otherwise transferred into and refined and augmented in the editor represented by the Edit step 14 shown in FIG.2. Concept also involves the determination of the various video, sound and other production assets 17 which need to be captured.

Creation of spatial video preferably involves first the layout of the environment which is to be shown in spatial video. This layout is an important part of the spatial video script 18. Laying out or designing the environment involves creating one or more environmental scenes or navigational units. These scenes or navigational units are map-based rather than temporal, although elements within the scene may have logical or temporal

attributes. The scenes may be derived from an existing structure or topography or created on a soundstage or in computer graphic renderings. However, a principal task to be accomplished in the concept stage is the inputting of the topography of the scenes into the editor.

5           The scenes or navigational units of the spatial video production can be virtually any imaginable environment, such as rooms, the entire interior of a house, a floor in a multi-floor building, a level in a ship, or space station, a chamber or set of chambers in a cave or mine, a lake spotted with islands or a forest glade. All spatial video scenes have certain common characteristics relevant to the invention. For greatest realism of navigation, the environment  
10       should be one which lends itself to navigation along a number of natural paths. It should also have environmental features with which the user can interact, entrances, path exits as well as scene exits, path intersections and map-based connections with plot elements and with other navigable units. The creation and characterization of these additional scene characteristics is a further task to be initially accomplished during the concept stage.

15           Just as in the concept stage of movie or television productions, a map-based spatial video script 18 may call for use of an existing location, such as the rooms and grounds of a castle or mansion, or the script may require the creation of rooms on a soundstage. Further, as is increasingly common in television and movie productions as well, time based navigational units can be rendered with computer graphics rather than being physically  
20       created. Objects, people and the like can be inserted into a spatial video production using known blue screen techniques and the like.

          The creation of the spatial script is substantially assisted by the authoring support provided by the spatial video editing system 14 (FIG. 2) of the present invention. In particular, the editing system allows the author to place the script into the map-based editor  
25       of the edit system 14, test and verify the scene and object relationships, and to verify that multiple-path problems have been identified and solved. Unlike a linear screen play where the scenes are always played back in the same order, the scenes in an interactive spatial video production might never be played back in the same order twice. Virtually every scene is inter-related with every other scene within the same act. In the present invention, the author  
30       can draw the outlines, proposed paths, hotspots, receptacles and relevant environmental features of the spatial scene in the map editor and use the editor to assist in final conception

of the project. Thus there is significant feedback between the Concept 12 and the Edit 14 steps of the process. In the map-editor the author can use text substitutes for missing components and for missing logic elements, can define relationships and model all of the “what-ifs” and “if-then’s” of the final production. Once this is done, the production can be test played with text as the medium. The text can be changed or replaced at any time and other media can be added. Thus the Edit 12 step looks to the Capture 13 step and the captured assets stored during that step to verify that the necessary assets have been captured. Where such assets are not among the stored assets, Edit 14 generates capture requests for such missing assets.

For example, in the initial spatial video script 18 inputted into the edit system of the Edit 14 step, the interactive multimedia author may have described a scene which would be represented in the edit system as a text statement and flagged as incomplete with a note that further design must be done before the scene can be filmed. Later, an art director may substitute part of that text with a sketch and the costume designer may replace the sketch with a different sketch showing a character in costume. These substitutions will update the representation of the missing final asset. A script writer may associate a particular spoken line with the character. Subsequently, the sketch may be replaced by a video clip accompanied by an audio clip. Ultimately, all the text and sketches may be replaced by video clips, logic elements, and other media components as they become available. It is one of the advantages of the present system that, during the edit process any one with access to the spatial video map based editor of the edit system could click on the area where the scene is to be and see the scene or its earlier stage representation.

The edit system of the present invention is a powerful authoring tool for creating and refining the spatial multimedia script. Just as scenes or dramatic elements can be sketched out and defined in the editor prior to the capture of the related asses, hotspots, receptacles, and objects can be planned, defined, and placed in the environment of a particular act and scene, and experimented with at any stage of their development. This allows the authors and editors to test and refine the logic elements of the script before capture of the assets. At any time during this concept step, the production can be test played, perhaps partially with text and partially with video and modified to make the production more appealing to the end user of the multimedia production. For example, the script of a video game might call for an

elevator to "take the player to a safer level" when an elevator call button is clicked on with a mouse. Further, the "safer level" can be determined in real-time when the player clicks on 1, 2, or n after the elevator arrives in answer to the call button. This action creates a multiple-way branch which can be tested for action and for completeness (no missing elements). By  
5 test playing the game during the authoring step and possibly before any video has been included in the production, missing elements can be identified and additional shooting can be scheduled as necessary. These capabilities help the author solve the multi-path problem and assure that there are no dead-ends and that all needed or desired hotspots, receptacles and objects are available for the production.

10 Equally important, however, is that all of these activities are done with a spatial orientation and with the help of the "map-based" authoring and editing means, about which more will be discussed later in the Description of the Invention. In practice, the script becomes the storyboard which becomes the layout of the environment, which becomes, ultimately, the production.

15 As stated above, the spatial video script, although primarily map-based, may still have time sensitive components. For example, a bookshelf may convert to a door when the user discovers a button or solves a puzzle. Further, a lamp could be "clicked" on by the user without activating the lamp hotspot unless the user has lamp oil in his inventory. Thus, elements of the spatial script will have conditional characteristics which must be associated  
20 with them in the logic of the production. Further, the production may include conventional dramatic sequences which are subject to conventional scene scripting, shooting (or animation) and editing. It is emphasized that the spatial video environment of the invention expands the developer's options rather than contracting them. Consequently many pre-existing, conventional multimedia components are capable of being integrated into spatial  
25 video productions with little change, albeit with enhanced effectiveness, which is an additional feature and benefit of the system of the present invention.

While the incorporation of these known production elements into a unique spatial video production is within the scope of the invention, this disclosure will emphasize the novel elements of the scripting process, rather than the novel applications of old techniques.  
30 These novel elements relate primarily to environmental or map-based constraints and advantages inherent in the use of spatial video.



Once the map-based script has been developed sufficiently during the concept stage to identify a room which is to be captured on video and the paths through the room on which navigation will be allowed in the production, the set can be built and the spatial video for that room captured.

5

### Capture Step

The concept step 12 of scripting identifies many components which must be "captured" by the system for use in the spatial video production. Of course conventional assets, such as dramatic video segments, musical performances, voices, environmental  
10 sounds, bitmaps and the video production tasks associated with any traditional movie or video production, such as shooting film footage elements, taping of video elements and recording of audio elements must also be performed and captured in the capture step 13.

According to the process of the invention, spatial video is created by first creating the environment in the concept 12 and edit 14 stages, then "capturing" 13 the various navigable  
15 units of the environment. While the system has great advantages in efficiency when used to capture physically existing environments, it also provides considerable advantages, particularly at runtime, when some or all of the environment is comprised of rendered scenes.

Central to the concept of spatial video is the use of video clips captured by moving  
20 the camera through the environment to be captured along the natural paths set out in the environment. The natural paths must be captured on video with sufficient total horizontal field of view to capture all of the visual information which a person might view when negotiating the natural paths in different directions and with different head positions.

The traditional tasks required for productions outside the invention are augmented by  
25 additional tasks attributable to the spatial video nature of the production. Of the four tasks discussed immediately below, only the fourth is necessary to create a spatial video production. The first three offer significant advantages and efficiencies in navigation and editing as well as smoother transitions. The four additional capture tasks are as follows:

1. Optically enhancing the viewed scene while video taping (or filming) for the  
30 production,

2. Recording of the camera position and angle while taping or filming for the production,
3. Recording of the topography of the environment for the production, and
4. Using special camera path techniques during video taping or filming.

5 As the techniques used for conventional elements are well known in the art, the capture portion of this disclosure is focused on the capture of navigable, spatial video. We will now discuss these four additional tasks of the capturing step.

### Optical Enhancement

10 While this step is not necessary in practicing the invention, the decision whether to use optical means to enhance the scenes being captured is preferably made prior to filming. The decision can be postponed for rendered scenes where the graphics can be more easily adjusted later.

15 Giving the end user of the production (the player) control over panning of the scene being viewed allows for increased interaction with the interactive multimedia video and thereby gives the user a greater sense of "being there." To provide the user an ability for panning requires that there be more picture captured and available than what is presented on the screen at any one time. In most environments being navigated from the point of view of a person on foot, horizontal panning is highly desirable, while vertical panning is less  
20 important. Consequently, it is desirable to allow for more undisplayed picture width than height. However, the same principals can be applied in environments where height panning is desired.

One way of increasing the captured horizontal width is to use an anamorphic lens. That is, a lens that produces a different magnification along horizontal lines in the image  
25 plane than along vertical lines in the image plane. More particularly, such a lens incorporates a cylindrical element in which the image is distorted so that the angle of coverage in a direction perpendicular to the cylinder is different for the image than for the object. The preferred lens is an anamorphic, cylindrical lens which compresses the horizontal image features 2X relative to the vertical features. Other lenses which could be  
30 used would be conventional wide angle lenses, fish-eye lenses and the like. Yet another

possible means for capturing extra picture width and/or height might be the use of multiple cameras mounted in tandem either horizontally or vertically, or both, and then stitching the captured scenes together during the editing phase. It is duly noted that such stitching is not a trivial task and achieving a seamless appearance in the results would require substantial skill and/or technique. However scene stitching techniques are well-known in the art and are included in Apple's QuickTime VR program.

Use of such lenses creates a distorted picture, which, when corrected provides a larger, stretched frame. Where the capturing techniques create horizontal compression, such as by use of an anamorphic lens, the captured image can be stretched under computer software control to remove or diminish the distortion. Further, even when using non-distorting lenses, some computer-controlled stretching can be accomplished without noticeable deterioration of picture quality, any and all of these methods allow for the provision of a larger scene than can fit on the screen. The user is then allowed to control, through panning, which segment on the wide screen is displayed. These means of capturing and stretching gives the ultimate user (game player) the perceived ability to turn his/her head a certain amount (pan the view) within the environment within the same frame of video.

In the preferred embodiment, the environment is captured using a wide angle lens, preferably a cylindrically projecting anamorphic lens with a 90 degree horizontal field of view. A non-anamorphic lens would yield about a 45 degree field of view. Thus, use of the preferred lens results, after correction, in twice the horizontal field of view of an ordinary lens.

There are three distinct types of image distortion present in the preferred embodiment of the spatial video system: Horizontal (anamorphic) compression; cylindrical projection distortion; and distortion caused by spherical imperfections in the cylindrical projection. Horizontal compression is introduced intentionally thorough the choice of an anamorphic lens to capture more scene width to support horizontal panning during playback, or to allow playback in a letterbox format. It is usually introduced by the optical elements of the anamorphic lens, but can also be introduced post capture or by available optical simulating algorithms if the scene is rendered. For post capture introduction, a "normal" (spherical) wide-angle lens is used to capture the video. It must capture 90 degrees of width. Since the lens is not anamorphic, it will also capture twice the needed height. The video is

subsequently edited to discard half the captured scan lines, and to expand the remaining scan lines so that they fill the frame. The resulting image is essentially the same as the one captured with the anamorphic lens, except it will contain less vertical detail. The reduction in vertical resolution is usually masked by the MPEG encoding which greatly reduces both vertical and horizontal resolution anyway. The horizontal distortion is dealt with at playback time when the picture is stretched horizontally by the MPEG player.

Distortion is also introduced by the use of a cylindrical projection lens (or lens simulation for rendered video). This distortion gives us scenes which fit together seamlessly to produce smooth panoramas. The scenes will appear to be somewhat distorted, however.

For example, a straight horizontal line near the top of the scene will appear to curve downward toward the left and right edges of the scene. (See text accompanying **FIGS. 3a, 3b, 4a and 4b**. In the preferred embodiment, there is no correction of this type of distortion, however it can be removed at playback time by expanding columns of video pixels in each frame, with greater levels of expansion applied to the columns near the left and right edges of the frame. For such correction not to interfere with smooth transitions, the correction must be done on-the-fly during playback, at least during transitions, or there will be unrealistic bends at the seams where frames join. See text accompanying **FIGS. 3a, 3b**.

A final type of distortion is spherical distortion which is due to inherent spherical imperfections in the commercially available cylindrical lenses. Rendered video can be free of this type of distortion. In a pure cylindrical projection, all vertical lines in the scene would show up as vertical lines in the image. The spherical distortion present in the lens results in vertical lines near the left and right edges of the scene bowing outward. We correct for this type of distortion by using standard video special effects editing equipment, as discussed below.

Use of the preferred wide angle anamorphic video allows for limited instantaneous navigation of the environment by panning to the non-displayed portions of the bitmap. As will be discussed below, use of the anamorphic lens also allows for realistic transitions between adjacent frames, provided the frames are shot where the nodal point of the camera lens is rotating about the same vertical axis point on the natural path. Where the spatial video along a particular path is taken by a plurality of camera trips along the path, as it is in

the preferred embodiment, certain precautions must be observed to ensure that the camera nodal point always passes through the same points as it traverses the path.

In order for the transitions function of the present invention to provide realistic transitions it is important that the seam between the edges of the FROM frame and the TO frame is not obvious. Various techniques can be employed for getting good seams. For example, the edges of the FROM and TO frames could be intentionally blurred to hide discontinuities, or these edges could be manually edited to match better using standard computer-based image manipulation techniques. In the preferred embodiment which uses a cylindrical projecting, anamorphic lens, the nodal point must be at the same position at a given point on the path regardless of the direction the camera is facing to allow for seamless transitions as discussed below. Under these preferred circumstances, the FROM and TO frames represent adjacent portions of a cylindrical projection image.

A comparison of **FIG. 3a** and **FIG. 3b** demonstrates the advantages of using the preferred cylindrical (anamorphic) lens for capture. Paired cylindrical projection images produce a more natural spatial video transition than a pair of rectangular projection images. Both **FIG. 3a** and **FIG. 3b** represent images with a total horizontal field-of-view of 180 degrees with the same center of projection. **FIG. 3a** is a pair of 90 degree rectangular projection images placed side by side, while **FIG. 3b** illustrates a pair of 90 degree cylindrical projection images placed side by side.

Imagery may assist in understanding the benefit of cylindrical projection technology for transitions. Imagine being at the center of a large transparent cylinder whose axis is vertical. Now imagine replacing this transparent cylinder with another of the same size, but this new cylinder is painted with a panoramic image that appeared to the eye to be identical to the view through the transparent cylinder. If one takes any portion of this cylindrical panorama and lays it flat, one has a cylindrical projection. Vertical straight lines remain straight, but horizontal straight lines appear curved, except for those at the "horizon" i.e., the circle formed by intersecting the cylinder with a horizontal plane at the same height as the viewer's eye. **FIG. 4b** illustrates a cylindrical projection of a grid of lines in a plane perpendicular to the gaze vector. By contrast, in a rectangular projection straight lines in any direction remain straight. If two cylindrical projection images are placed side-by-side, they form a perfect larger cylindrical projection if, and only if:

1. The center of projection is the same (nodal point of camera lens is in the same point in 3-D space)
2. The center axes of the cylinders coincide
3. The vertical field-of-view is the same and the horizons line up
- 5 4. There is no overlap or underlap at the common edge.

If the two images are the FROM and TO images of a transition, such as those disclosed in the co-pending Special Effects Applications incorporated herein by reference, the result will be a seamless transition. If the same is attempted with two rectangular projection photographs taken from the same point the straight edges that straddle the two photographs bend at the seam between the images as illustrated in **FIG. 3a**.

In practice the anamorphic lens used in the preferred embodiment of the system does not yield a perfect cylindrical projection, but possesses some spherical distortion. When the images are scaled horizontally such that the horizontal field-of-view at the horizon is 90 degrees, the field-of-view along the top and bottom of the frame is slightly greater than 90 degrees. Vertical lines bow outward as depicted in the grid picture of **FIG. 4a**. The amount of bowing in **FIG. 4a** is exaggerated to illustrate the effect. In the preferred embodiment, therefore, prior to encoding, the video is processed with digital video effects equipment to more closely approximate a true cylindrical projection as discussed above. **FIG. 4b** illustrates how the **FIG. 4a** grid picture would look after digital video processing. Notice that the vertical lines are substantially corrected, resulting in a more accurate cylindrical projection.

Frames taken with the preferred cylindrical anamorphic lens are distorted horizontally to fit a 90 degree (horizontal) field of view on the frame which would yield a 45 degree field of view with an ordinary lens. Images captured using such a lens, where the vertical field of view is set appropriately and the horizontal distortion corrected (by 2X horizontal stretching), result in a picture having twice the usual width but still having the usual height.

In wide screen movie productions such as Cinemascope<sup>7</sup> and Panavision<sup>7</sup> productions which use an anamorphic lens technology, a corrective lens is used on the projection equipment which reduces this distortion. In the preferred process of the present invention, digital effects are employed to make similar optical corrections to each picture

after the anamorphic image is placed on video, but prior to MPEG encoding. In the preferred embodiment, a sine wave distortion is applied using the DPM-1 Kaleidoscope brand digital video effects machine available from the Grass Valley Group. The exact setting of the controls is determined empirically by manipulating a picture containing vertical lines near the left and right edges. The amount of sine wave correction is adjusted until these vertical lines appear to be straight, thus more closely approximating a true cylindrical projection. The images are then scaled horizontally such that the field of view at the horizon is 90 degrees and sent to the MPEG encoder.

To obtain the best transition results the center of projection should be the same for the FROM and TO frames. This means that the nodal point of the camera lens should be located at the same point for the frames that will become the FROM and TO frames in the transition. Another way to say this, is that the camera must be placed in a position when filming the two transition frames to avoid parallax error.

This has special implications for roundabout transitions, which are transitions where the user is able to rotate at a single position in 3-D space. In the spatial video of the preferred embodiment of the present invention, the user can rotate 360 degrees by doing a transition, for example, from a forward to a left facing frame, then to a rearward facing frame, then to a right facing frame, finally back to a forward view frame. For the most realistic transitions, all four heading pictures taken at a roundabout point on the path should be taken with the camera lens nodal point at the same 3-D coordinate.

The requirement of the preferred embodiment that the center axes of the cylinders coincide usually means that the camera must be level at all intersections and roundabout points. It is possible to satisfy this requirement without being level if, for example, the FROM frame was taken with the camera pitched down by, say, 10 degrees and the TO frame was taken with a camera roll angle of 10 degrees. This would be the case if the center axis was tilted 10 degrees from the vertical. (This example assumes a 90 degree field of view as in the preferred embodiment).

For the best matched transitions the vertical field-of-view should be the same and the horizons should line up. This means that the vertical framing adjustments during the telecine process should be the same for a given pair of FROM and TO frames. Normally, the vertical framing will be constant for all spatial video in a production. If a different lens was used to

film the FROM frame as compared with the TO frame, one must compensate for any optical differences.

The requirement that there be no overlap or underlap at the common edge means that the camera paths should intersect at an angle corresponding to the horizontal field-of-view of the resulting MPEG video bitstream (90 degrees in the preferred embodiment). Although the intersecting paths should, therefore, be perpendicular at the point of intersection in the preferred embodiment, there is no requirement that the paths be straight. However, when the environment is to be captured by multiple passes of the camera on a single path, as in the preferred embodiment, it is important that the paths be set out in a way to facilitate accurate retracing of the path.

The technique set out immediately above, when coupled with the transition processes disclosure from the Special Effects Applications which are incorporated herein by reference, provides a full disclosure of the transition techniques of the present invention, except for the use of the edit system to identify and edit such transitions which is discussed below.

#### Camera Tracking

It is not necessary to track the camera position and angle when filming or taping scenes to be used in spatial video or to associate position and bearing data such as the data which camera tracking would generate, with the frames of the video clips taken in the environment. Neither is it necessary to record position data for the outlines of important objects in the environment. However, capture and use of positional data for video frames, object locations and outlines can enhance the efficiency of the production in a number of ways.

First, positional data associated with clips and frames as well as positional data associated with various significant features of the environment being captured can be used in conjunction with a top down map of the spatial video environment to plot camera tracks and features. Further, the availability of positional data on video clips sought to be imported into the production provides the editor the ability to quickly identify the relevant clips, frames and features needed to quickly create intersection and roundabout transitions as well as hotspots and receptacles. One can browse through the plots of the camera paths traversed in creation of the various available video clips and more quickly identify the clips and frames



associated with particular areas of the environment by looking at the positional data, more efficiently than by playing the video and attempting to visually recognize the clip and frame associations. Of course, visual verification of the frames located by use of tracking data is always desirable.

5           A second advantage to the use of camera position tracking and the positional data generated hereby is in keeping the camera on the proper path. As mentioned briefly above, the ability to make smooth transitions between adjacent video clips in navigation along and between natural paths is an important aspect of the present invention and crucial to the impression of immersion in the environment which spatial video provides. To allow the  
10       system to create such smooth transitions which allow user to turn further than the width of the large anamorphic frame and to switch seamlessly from path to path within the environment, the nodal point of the camera lens must follow precisely the camera path being traversed regardless of the direction the camera is pointing. As mentioned above, the “camera path” is a widthless line through the 3-D space of the environment. Use of tracking  
15       data at the time of filming can assist in maintaining the correct nodal point position for all of the clips taken on a single path (forward facing, rearward facing, left and right).

          A third advantage to the use of positional data is to assist in determining frame accurate intersection locations on the video clips of the intersecting clip groups. The location of an intersection of two camera paths using positional data can be accomplished by  
20       overlaying the various camera paths and identifying the coordinates of the intersection. Knowing the coordinates of the intersection, the frames in each clip with spatial coordinates closest to those of the intersection can be quickly identified and retrieved. These frames can be visually checked and fine tuned using visual means to select the FROM and TO frames with the best match, and the best matched frames then specified as the FROM and TO  
25       frames for the desired transition. The positional data is also helpful in identifying the best frames for roundabout transitions between clips in the same clip group.

          A fourth advantage is in placing “hotspots” or “receptacles” associated with significant features of the environment. Where the 3-D coordinates of the feature are known either through manual measurement or through location sensing means, the positional data  
30       associated with such features can be compared with the positional data of the various clips and frames. This will efficiently allow the project editor to identify camera clips and frames

on which the feature is likely to appear in order to superimpose hotspots or receptacles on such features.

Thus, although spatial video productions can be created without access to positioning data, the availability of such data creates many efficiencies in capturing and editing. Consequently, use of a position encoding device to identify the position of the camera in space with six degrees of freedom greatly assists in frame location and matching during the editing process and, therefore, is an important innovation in its own right. It is also an advantage to identify significant features of the environment by their 3-D coordinates. These two tasks are accomplished using position encoding.

The position encoder task can utilize a number of means or combinations of means. One means is to use ultrasonic transducers which enable one to calculate the camera position by pinging a pulse from a transmitter co-located with the camera and measuring the time of flight between the transmitter and multiple stationary receivers which have been placed strategically within a camera tracking area. A camera tracking area is defined as that area which can be covered by one placement of the fixed transducers. Multiple camera tracking areas are combined to make up the total environment of a production.

Another means to achieve camera position encoding could be a measurement system based on an eye-safe laser strobe and networked sensors system which records when the rotating beacon arrives at each sensor. Data from this approach would then be processed by a computer to determine the location of the camera within a pre-determined camera tracking area.

Other methods to determine position of the camera could include any mechanical, sonic, radio frequency or optical means for tracking movements of the camera base, mounting head, and stand provided such means produces a required degree of accuracy and repeatability, or by using a computer motion controlled camera system.

Camera Positioning System. The positioning system as used in one preferred embodiment of the present invention consists of a moving remote sensor co-located with the camera whose position is to be determined, a number of stationary receivers, an ultrasonic controller unit which outputs the data and a camera tracking control computer. The positioning system is available from Intelligent Solutions, Inc. of Marblehead, MA. The camera sensor uses an electronic compass and a two axis clinometer to measure azimuth,

pitch, and roll. This information is ultimately transferred to the camera tracking control computer. Also located in the camera sensor is a transmitting ultrasonic transducer. This transducer emits a burst of ultrasonic energy (a "ping") upon receipt of a start command from the central computer. This ping is detected by the receivers which then send a signal back to the ultrasonic controller unit. The interval between the ultrasonic transmit start time and the time of the received signal from the receivers provides a measure of the distance from the moving transmitter to the fixed receivers.

A preferred embodiment of the present invention employs the ultrasonic position tracking system from Intelligent Solutions, Inc. and the push-pull positioning process of this invention which is described below to determine the X, Y, and Z coordinates of the camera in the navigable space.

**FIG. 5** illustrates the components of the camera tracking apparatus according to the preferred embodiment of the present invention. The apparatus includes a camera sensor means **24** co-located with either the motion picture camera **26** or the video camera **28**, and connected to an ultrasonic controller unit **22**. Inside the camera sensor means **24** is an electronic compass **30** and an ultrasonic transmitter **32** attached to a sensor base. Also, up to eight ultrasonic receivers **36** may be connected to the ultrasonic controller unit **22** by means of cables **38** and channels **40**. In the alternative, radio links may be used to make set-up easier and more flexible. Ultrasonic receivers **36** are strategically placed at various locations throughout the environment with sufficient distance between each receiver and the ultrasonic transmitter **32** and sufficient distance between each of the several ultrasonic receivers **36**, to permit a time of flight diversity to be measured.

Also connected to the ultrasonic controller unit **22** is an ultrasonic hand-held wand **42**, said connection being made by cable **44**. The ultrasonic hand-held wand **42** is also an ultrasonic transmitter. It contains a button **43** which, through communications cable **44** or alternatively through a radio link, is connected to the ultrasonic controller unit **22**. Thus, the ultrasonic wand can be used in a manner analogous to that of the camera transmitter to determine, when actuated adjacent to such item, the X, Y, and Z coordinates of any item in the environment.

Ultrasonic controller unit **22** communicates with camera tracking control computer **46** by means of serial communications cables **48** and **50**. Camera tracking control computer

46 also accepts serial input via a third serial port (not shown) from the time code translator 54 (FIG. 6 and FIG. 7).

Ultrasonic Controller Unit. Ultrasonic controller unit 22, upon command from camera tracking control computer 46, causes ultrasonic transmitter 32 to emit a ping. In the case of the hand-held wand 42, the command is triggered initially by the depression of the button 43 on the wand 42.

The ultrasonic ping from the camera mounted transmitter 32 or the hand-held wand 42 travels to the receivers 36. Some filtering of the received sound occurs in the receiver, which reduces the receiver's output to a binary "yes" (receiving 40 KHz sound) or "no" (not receiving 40kHz sound). This binary yes/no goes to the ultrasonic controller unit 22 through the cables 38 and the connecting channels 40. The controller 22 uses a digital signal processor (not shown) and other circuitry (not shown) to sample the inputs and measure the duration of the yes signal for each reporting receiver. The duration measure is used as a filter to eliminate false yes reports from the ultrasonic receivers 36. The controller's 22 processor also measures the time delay from the start of the transmitted ping to the leading edge of the received signal. This delay measure is proportional to the distance from the ultrasonic transmitter 32 or wand 42 to the reporting receiver 36. Finally, the controller 22 sends the time-of-flight data for each receiver to the camera tracking control computer 46 over an RS-232 serial cable 12.

While many acceptable communications protocols are known in the art which are easily adaptable to this task, in the preferred embodiment the ultrasonic controller unit 22 available from Intelligent Solutions, Inc., receives the following commands from and provides the following responses to camera tracking control computer 46 in performing the above summarized tasks:

#### Commands -

*Trigger Command:* TFFFF<CR>, where:

T indicates the trigger command which causes an ultrasonic ping to be emitted from the camera sensor 24, and,

FFFF is a 4 digit hexadecimal number that defines the length of time, in units of sample time, that the ultrasonic controller unit 22 is to wait for signals to be returned from

ultrasonic receivers **36** before sampling is stopped. The sample time is 29.053 microseconds in the preferred embodiment.

<CR> is carriage return.

5           *Duration Command:*           DFF<CR>, where

D indicates the duration command, and

FF is a 2 digit hexadecimal number that defines duration, in units of sample time, of ultrasonic pings emitted from the camera sensor **24**. In the preferred embodiment, the ping duration is normally set to 172 sample times, or 5 milliseconds.

10

*Minimum Width Command:* MFF<CR>, where

M indicates the minimum width command, and

FF is a 2 digit hexadecimal number that defines the minimum width, in units of sample time, for a valid received ping. When the binary signal from an ultrasonic receiver **36** to the ultrasonic controller unit **22** is active for at least the specified minimum number of sample times, it is deemed to indicate a valid ping, and a range data response is sent to the camera tracking control computer **46** as described below. Because the ultrasonic receivers **36** may respond to spurious noises in the camera tracking environment, setting the minimum width too low would cause these spurious noises to be improperly reported as pings. Setting the minimum width too high would cause real pings to be rejected. In the preferred embodiment, the minimum width is normally set to 135 sample time units, or 3.92 milliseconds.

15

20

*Intensity Command*           IFF<CR>, where

25

I indicates the intensity command, and

FF is a 2 digit hexadecimal number that sets the sound amplitude to be transmitted by ultrasonic transmitter **32** and ultrasonic hand-held wand **42**.

*Set Command:*           SFF<CR>, where

30

S indicates the set command, and

*FF* is a 2 digit hexadecimal number that sets a mask to enable sampling a channel **40**. If the corresponding bit is set, the channels is enabled. If the bit is cleared, the channel is ignored even though it may have received a signal.

## 5 Responses -

*Range Data:* RX:FFFF<CR><LF>, where

R indicates the range data response,

X is a digit from 0 through 7 indicating which channel **40** has detected a ping, and

FFFF is a 4-digit hexadecimal number that indicates the time delay in units of sample time, from the start of the transmitted ping to the leading edge of the received signal in units of sample time.

*Termination Character:* Z<CR><LF>, where

Z indicates that the sample interval, or time-out period is over. Additional commands from camera tracking control computer **46** should not be sent to ultrasonic controller unit **22** until the termination character is sent indicating that ultrasonic controller unit **22** is ready.

Ultrasonic Handheld Wand. The ultrasonic handheld wand **42** is a manually triggered ultrasonic transmitter manufactured by Intelligent Solutions, Inc. of Marblehead, MA that can be used to verify system operation, assist in measuring the position of receivers **36**, and assist in measuring positions of path intersections, turns, corners, and other features within the environment as input to determining topography of the environment. The wand transmitter operates in a fashion similar to that of the camera sensor transmitter. The only differences are that the wand emits a ping in response to depression of the wand button **43**. Depressing the button also signals the controller unit **22** over cable **44** that a wand ping was started.

25 Ultrasonic Receivers. Ultrasonic receivers **36**, also available from Intelligent Solutions, Inc., each contain a transducer and a 40 KHz detection circuit. Each receiver sends a binary signal over a cable **38** to the ultrasonic controller unit **22** indicating whether or not they are receiving 40 KHz sound at any given instant.

30 Camera Sensor. Camera sensor **24** contains compass **30**, ultrasonic transmitter **32**, a cone (optional and not shown) and a circuit board which is contained in the sensor base **34**. The cone (optional and not shown) converts the ultrasonic beam from the ultrasonic

transmitter 32, into an omni-directional pattern in the horizontal plane. Such a cone may be helpful to obtain more reliable receipt of the transmitter signal where the receivers are in horizontal orientation with the transmitter. The cone is not needed where the transmitter's 32 natural dispersion pattern provides sufficient signal strength to the receivers.

5           The circuit board in the sensor base 34 receives RS-422 differential signals on cable 66 from ultrasonic controller unit 22 and converts them to single ended RS-232 levels for interfacing with compass 30. The reverse operation is performed for signals from compass 30 to ultrasonic controller unit 22 which, after conversion to RS-422 signals are sent to the ultrasonic controller unit 22 along the same cable 66.

10           The circuit board in the sensor base 34 also receives the ultrasonic burst signal (preferably 5 milliseconds of 40 KHz) from ultrasonic controller unit 22 on cable 66 and converts it to a single ended signal for driving ultrasonic transmitter 32.

Compass. Electronic compass 30 is preferably the TCM2 Electronic Compass Module from Precision Navigation, Inc. of Mountain View, California. The compass is  
15           mounted in camera sensor 24 and outputs heading, pitch, and roll measurements via cable 66 to ultrasonic controller unit 22 which, in turn sends the measurements to camera tracking control computer 46 via an interface conversion function described below. Compass data is derived from the manufacturer's proprietary tri-axial magneto-inductive magnetometer system and a biaxial electrolytic inclinometer, and contains no moving parts. The  
20           inclinometer permits electronic gimbaling for the magnetometers as well as providing information on pitch and roll angles. This information, fed to a microprocessor internal to the electronic compass module, allows the electronic compass module to mathematically correct for tilt of the unit and to provide continuous and automatic correction for hard iron distortion as the system is used. The compass 30 is activated by a start signal sent by the  
25           camera tracking control computer 46 along RS-232 serial cable 48, to the ultrasonic controller unit 22 where it is converted to an RS-422 signal and forwarded on cable 66 to the sensor base 34 portion of the camera sensor 24. The sensor base 34 reconverts the signal to an RS-232 signal and passes to the connected compass 30.

30           Once started, the compass module sends RS-232 data to the connected sensor base 34 where it is stepped up to RS-422 and sent to ultrasonic controller unit 22 by cable 66 over an RS-422 interface using ASCII coded characters at a baud rate determined by the user. In

ultrasonic controller 22, the signal is converted from RS-422 back to RS-232 and forwarded over cable 48 to camera tracking control computer 46. The standard output format may be configured to provide all of the sensor data parameters available, or only those parameters required by the user. Assuming all parameters are to be sent, the standard output format is:

\$C<compass>P<pitch>R<roll>X<Bx>Y<By>Z<Bz>T<temp>F<errorcode>\*checksum<cr><lf>

Example: The compass module will return the following:

\$C328.3P28.4R12.4X55.11Y12.33Z18.43T22.3E001\*checksum<cr><lf>

under the following conditions:

compass heading = 328.3 deg. (true or magnetic, depending on configuration)

pitch = 28.4 degrees

roll = 12.4 degrees

Bx = 55.11 uT (x-component of magnetic field)

By = 12.33 uT (y-component of magnetic field)

Bz = 18.43 uT (z-component of magnetic field)

Temperature = 22.3 degrees (F/C depending on configuration)

E001 = Distortion flag is raised - magnetic anomaly nearby

Any parameters not enabled by the user are not included in the output word, e.g.

\$C328.3T22.3\*checksum<cr><lf> for compass and thermometer information only. For further information the reader is referred to the TCM2 Electronic Compass Module User's Manual, Revision 1.011, March 17, 1995 which is available from the manufacturer.

The ultrasonic TOF data captured by the camera tracking control computer 46 and the compass bearing data must be converted into position and orientation (6-D) coordinate data and associated with the specific video frames as they appear in the MPEG video bitstream to be most helpful in the spatial video production process. This is accomplished through converting the TOF data into positional data and correlation the positional data with the SMPTE time codes.

The camera tracking control computer 46 is discussed in more detail below in the discussion relating to FIG. 6 and FIG. 7. While FIG. 5 shows two serial inputs to the



camera tracking computer, the compass data and the ultrasonic data inputs, not shown in **FIG. 5** is the third serial input which comes from the time code translator. The source of the time code data varies depending on whether the digital video assets to be captured originate as motion picture film or are originally shot as video. We will first discuss the process of correlation of time code data with video frame data and with camera position data where a motion picture film camera is used.

Correlation of Film Frames and Camera Position Data. The method for correlating the camera tracking data to frames of an MPEG video bitstream is slightly different for a film camera vs. a video camera. But in each case, a SMPTE time code signal is integral to the process. The *hour*, *minute*, *second* and *frame* components of the SMPTE time code have a one to one correspondence with each frame of film or video.

Referring to **FIG. 6** a time code translator **54** is connected to camera tracking control computer **46**, by means of cable **52**. The time code translator **54** translates SMPTE time codes into an ASCII format which can be communicated to a PC, such as the camera tracking control computer **46** through a serial connection **52**. As **FIG. 6** illustrates capture using a motion picture camera, the time code translator **54**, receives the time code signal through its connection to a master SMPTE clock **60**. The master SMPTE clock is jam-synched to the time code writer portion **62** of the motion picture camera **26** using the techniques well known in the art. Further, a smart slate **64** may also be jam-synched with the master SMPTE clock **60** to be used as a backup to the time code writer **62**.

The output of the camera tracking control computer **46**, is compass and TOF data. This data is time-stamped as received by the computer using the computer's internal clock. The time-stamped data is stored in the CTJ (camera tracking journal) file **68**.

The motion picture camera **28** is equipped with a time code writer **62** which optically exposes a time code on each frame of film as it is shot. Prior to filming the time code writer **62** is jam-synched with the master SMPTE clock **60**, thus the time code exposed on the film is the same as that being produced by the master SMPTE clock **60**. The master SMPTE clock **60** is also connected to the time code translator **54**, which in turn is connected to the camera tracking control computer **46**. The time code received by the camera tracking control computer **46** is also time-stamped using the computer's internal clock before it is entered into the .CTJ file **68**.

The .CTJ file data is converted into camera position data with the CAMTRAK program processing 70 which results in a .CPA (camera position ASCII) file 72. This data associates 6-D camera position measurements with SMPTE time codes. The .CPA position data is then entered into a clip position table in the master database file 74 of the spatial video edit system 14 where it awaits synchronization with the film after its conversion to video.

After shooting, the film is developed. The developed negative 76, with the optical SMPTE time code on it, is converted into video by use of a telecine machine 78 equipped with a time code reader 80, this device reads the optical SMPTE code off of the film during the telecine process. It is important to note that time codes associated with the various frames of the developed negative differ from the time codes of the corresponding video frames. However, with the use of the time code reader 80 a table can be constructed which provides for each frame, the film time code and corresponding video time code from the resulting video tape 82. This information is recorded in the sync point database 84. The time code data can also be created using a smart slate 64 as described below.

As film is typically shot at a rate of 24 frames per second, the frames component of the SMPTE time code for film repeatedly increments through the range of values from 0 through 23 (signifying 24 frames in each second). In contrast video is shot at approximately 30 frames per second, with the frame portion of the time code incrementing from 0 through 29. Thus film time codes will not correlate 1:1 with the converted video time codes, due to the use of the 3-2 pull down process to create 30 frame-per-second video from 24 frame-per-second film. In the preferred embodiment, the 30 frame-per-second video is converted to 24 frame-per-second MPEG during the encoding process by means of a reverse telecine process performed by the MPEG encoder. The telecine and reverse telecine processes and the time code correlation process are well known in the art and will not be further discussed here.

As the camera tracking control computer 46 is not directly connected to the camera 26, at the beginning of each camera run the camera operator verbally or visually notifies the camera tracking control computer operator that filming is about to begin and the computer operator signals the computer to begin tracking. The camera operator may also run film to test lighting and equipment while the camera is not in motion. Consequently, there will be time codes in the .CPA files 72 for which no film frames correlate, and there will be film

frames for which there is no positioning data. As the time codes in the .CPA files and those associated with the film will not match up exactly, it is the function of the sync point database to assist in eliminating superfluous camera tracking data and film frames by matching up the common master time codes.

5           The sync point database **84** and the camera position/time code table from the edit system database **74** are the inputs to the encoding assistant **86**. The encoding assistant **86** is a program which presents the user with side-by-side lists of clips from the sync point database **84** and the Edit system database **74**. These lists are sorted according to the master SMPTE clock **60** time code and clips are automatically paired up when an overlap of time  
10   code ranges is detected. The user then selects from the set of clips with paired sync point and camera position time code ranges to create a list of clips that will be encoded by the MPEG encoder **90**. This list is written to the encoding batch file **88** in a format appropriate for the particular MPEG encoder **90** being used. Along with the starting and ending video tape time codes (known as in points and out points), each batch entry includes a time code  
15   corresponding to the master SMPTE clock **60** time for the first frame of the clip as determined by the encoding assistant. The MPEG encoder **90** is instructed to use this time code for the *time\_code* field in the GOP headers of the bitstream.

When the MPEG encoder **90** is operated, it uses the video tape in and out points to automatically cue, start and stop the VTR (video tape recorder) **92**. When the encoded  
20   MPEG bitstreams **94** are imported into the edit system **14**, the *time\_code* field from the first GOP header is used to compute an offset between the first frame of camera position/orientation data and the first frame of MPEG video, which completes the correlation process.

An alternative way of associating the master SMPTE clock **60** with the film is to use  
25   an apparatus known as a smart slate **64**. This apparatus contains a crystal-controlled SMPTE time code clock like the time code writer **62**, and is synchronized to the master SMPTE clock with the same jam-synching method. The smart slate, with its illuminated time code display, is held in front of the camera for a brief period each time the camera is started. In this alternative, the video tape is viewed by a human operator after the telecine process and the in  
30   point out point and smart slate time code for each clip are recorded and entered into an alternative form of the sync point database.

Correlation of Video Frames and Camera Position Data. If a video camera is used instead of a motion picture camera, the process is simpler. As illustrated in FIG. 7, the video tape recorder portion 56 of the video camera 28 contains an internal SMPTE clock. This clock associates the time code with the video tape 82 and also provides a time code signal to the time code translator 54 through cable 96 which, in turn provides an ASCII version of the time code to the camera tracking control computer 46 over cable 52. Because the VTR portion 56 of the video camera 28 acts as the master time code clock and as the camera tracking time code source, there is no need for a sync point database. The encoding assistant 88 is able to generate the required batch files with just the .MDB input.

Use of Rendered Frames. If the MPEG bitstream originates with rendered images instead of film or video tape, then a .CPA file with virtual camera position/orientation information is supplied by the rendering process, and correlation is straightforward. The starting time code for a rendered clip is arbitrary, as long as the same time code is used in the .CPA file 72 and the encoded MPEG bitstream 94.

Time Code Writer. The film camera 28 is outfitted with a Time Code Writer apparatus such as the AatonCode available from Aaton Camera, Grenoble, France. This apparatus contains a crystal-controlled SMPTE time code clock and it optically exposes the time code right on the film as the camera is operated.

Telecine Machine. The film is converted into video using a telecine machine such as the BTS Quadra Telecine. The time code reader used with the telecine machine is part of the Aaton Keycode System, also available from Aaton Camera, Grenoble, France.

Time Code Translator. In order to later associate the master SMPTE or VTR time code with the camera tracking data an off-the-shelf device called a "time code translator" is used to convert an SMPTE time code signal to an RS-232 format that is read over a serial port. An example of such a device is the "EASY READER I" product from Telecom Research, Burlington, Ontario, Canada. Time code translator 54 reads the 80-bit time codes of the Society of Motion Picture and Television Engineers (SMPTE) and the European Broadcasting Union (EBU). This code, when recorded on video or audio tape permits exact addressing of points on the tapes for precise editing, synchronization, dubbing, and splicing. For every frame of video, there is a corresponding Time Code Address. The address is an 8-

digit number representing HOURS: MINUTES: SECONDS: FRAMES where the FRAMES number represents the  $n^{\text{th}}$  frame recorded during the current SECOND.

Master SMPTE Clock. In FIG. 6, the master SMPTE clock 60 is preferably the Aaton OriginC manufactured by Aaton Camera of Grenoble, France. The time code writer on the camera is synchronized with the master clock using the standard jam-synching procedures. This synchronization is repeated at regular intervals, such as once every eight hours during filming to ensure that the clocks do not drift apart.

Smart Slate. Any smart slate is usable. The Smart Slate, DCODE TS-2 Time Code Slate from Denecke, Inc., North Hollywood, California is such a device. In FIG. 6, the smart slate 64 is used to display the time when a scissor-like bar (not shown) is shut or "clapped" to a display panel (not shown), thus allowing the displayed time to be captured on video tape or film. As discussed above, it is prudent to use a smart slate as a backup in case of malfunction of the time code writer or reader. Often a malfunction of the time code writer might not be detected until telecine time. Further the slate provides a way to verify the calibration of the time code reader.

Camera Tracking Control Computer. The Camera Tracking Control Computer 46 computer can be a standard PC (personal computer) clone with the addition of a multi-port serial card in order to accommodate the high volume of interrupt-driven serial port traffic. During filming the computer 46 runs a custom program called CAMTRAK that, in addition to controlling the Camera Tracking Apparatus, reads the Master SMPTE Clock via the time code translator 54. The camera tracking computer records and time stamps the time code information from the Master SMPTE Clock along with the other camera tracking data generated by the ultrasonic positioning system and the electronic compass.

The CAMTRAK program requires the following serial I/O capabilities:

1. Interrupt-driven receive with buffering
2. Support for COM1, COM2, COM3 simultaneously
3. Support for up to 38.4 kBaud
4. Ability to report hardware errors (parity, overrun, etc.)
5. Time stamping of first byte in a message

The PC's standard ROM BIOS serial communication routines do not meet these requirements, so CAMTRAK installs its own interrupt handler for serial I/O. For ease of

implementation as well as runtime performance, the multi-port serial card is configured such that each port is assigned its own interrupt. In order to facilitate subsequent correlation of the camera tracking data with the Master SMPTE Clock's time code, each byte received on each serial port is time stamped by the serial port interrupt handler. When CAMTRAK  
5 parses the bytes received from each device, the time stamp of the first byte of a message is saved with the message. The PC's timer interrupt increments the system clock tick counter at a rate of approximately 18.2 Hz. This is not sufficiently fine-grained to permit accurate correlation of all of the data. Therefore the serial port interrupt handler reads to most significant byte of the Programmable Interval Timer (either Intel 8253 or 8254) counter and  
10 combines this with the three least significant bytes of the system clock tick counter to create a time stamp with a resolution of 215 microseconds which is accurate enough for correlation.

Each time the camera is tracked, the time stamped messages from the Master SMPTE Clock and the Camera Tracking Apparatus are saved in a file with the extension .CTJ for CAMTRACK Journal 68. The CAMTRAK program's analysis mode is used to convert the  
15 .CTJ file into a .CPA (Camera Position ASCII) file 72. This analysis can be performed on the camera tracking control computer as the filming is occurring or by any PC at a later time.

The .CPA file has X, Y, Z, heading, pitch and roll values to completely define the camera's position and orientation for each frame of film as identified by the time code from the Master SMPTE Clock.

20 MPEG Encoder. The analog video sequences from the video tape 82 are encoded by an MPEG Encoder 90, such as the RTS-3000 MPEG Encoder from Sony Corporation. The encoder uses the video tape recorder's (VTR) inputs and outputs to automatically cue, start and stop the VTR. When source material is 24 frame-per-second film, such that the video was generated by the telecine machine using the 3-2 pull down process the MPEG encoder  
25 preferably performs a reverse telecine so that the resulting MPEG stream is 24 frame-per-second MPEG. When the encoded MPEG Bitstreams are imported into the editor, the *time\_code* field from the first GOP header is used to compute an offset between the first frame of camera position/orientation data and the first frame of MPEG video, which completes the correlation process.

30 Push-Pull Positioning Algorithm. Prior art algorithms for use in ultrasonic trilateration employ a least squares estimate and are valid only if the expected value of the

error of each measurement is zero. When tracking the ultrasonic transmitter over a large area containing line-of-sight obstructions, and where the transmitter while traversing a camera path is moving into and out of range of one or more sensors, the positional data generated using prior art least squares algorithms contained sufficient false signals and large errors to seriously affect its viability in a camera tracking system. One aspect of the current invention is a more robust method of determining location under such circumstances.

In order to solve the problems of false signals and large errors encountered when using prior art methods in environments containing line-of-sight obstructions and a transmitter moving into and out of the range of one or more receivers, the system disclosed herein employs a "push-pull" method which determines the position of the ultrasonic transponder by iteratively calculating and weighting force vectors using the TOF data collected by the ultrasonic receivers in the environment.

More specifically, the push pull method of the invention employs fractional force vectors computed from the collected ultrasonic TOF data to provide accurate positional data.

Assuming omni-directional transmission and reception for the ultrasonic transponders, a single ultrasonic TOF value defines a sphere about a fixed ultrasonic receiver, said sphere being the set of all possible locations for the transmitter. TOF values from a pair of receivers define the intersection of two spheres, which is either a circle, or, if the transmitter is collinear with the receivers, a single point. The sphere around a third receiver will intersect said circle at two points, in the general case. These two points are a mirror image of each other in the plane defined by the three receivers. The sphere around a fourth receiver will intersect only one of these points, unless the fourth receiver is coplanar with the first three. The sphere defined by the TOF value of each additional receiver would intersect the same point, if the values were without error.

The frequency of pings used in the preferred system is under user control. In most environments the preferred frequency is eight times per second in the environments used by inventor, with frame position data for frames shot between pings created by linear interpolation of position data calculated from the bracketing pings. However, the ideal frequency varies as a function of the acoustic characteristics of the environment and the speed with which the camera traverses the environment. The goal is to ping as often as possible in order to reduce the amount of interpolation used, especially if the camera is

traveling quickly, but to keep the ping frequency low enough so that ping echoes have faded between pings.

**FIG. 8** illustrates an embodiment of the position calculating process of the invention where the push-pull processing is not being done in real time, but in post-shooting processing. Thus the TOF values being accessed immediately after “start” are previously recorded and contained in a .CTJ file **68** (see **FIG. 6**). The process also adjusts TOF values for the inherent system delay (this adjustment step is not illustrated). According to the process of the invention, an initial assumed position of the transmitter is chosen and the distance from this point to each receiver is computed. This distance is compared with the TOF value between the transmitter and the receiver. Next, a three-dimensional “force vector” is computed based on the assumed position chosen for the transmitter for each receiver for the first “pings” TOF data. The calculated force vector either “pushes away” from the receiver (if the TOF value is greater than the computed distance) or “pulls toward” the receiver (if the TOF value is less than the calculated distance). The force vectors from all receivers are first weighted, as discussed below and then combined into a single vector, the value of the combined force vector is normalized as described below and the assumed position of the transmitter is adjusted by some fraction of this vector. Then the process is repeated until the magnitude of the “push” or “pull” of the force vector is less than a pre-chosen value which was chosen based on the required accuracy of the result.

Only a fraction of the vector is applied for each iteration in order to reduce the likelihood of falling into a false, local minimum. For the preferred embodiment the empirically chosen fraction is 2. A local minimum is a measurement point in which, due to anomalous acoustic properties of the environment, the ultrasonic trilateration system computes a combined force vector of zero, although in actuality the calculated point is not its true position.

The system continues to iteratively calculate 3-D force vectors based on the first ping TOF data and to update the assumed position of the camera or wand until the combined force vector is less than epsilon. In the preferred embodiment the empirically chosen value of epsilon is 1/100 centimeter. When computing the mobile transmitter position for a series of ultrasonic pulses, if the interval between the start of one ultrasonic pulse and the start of the next and the velocity of the mobile transmitter are both sufficiently small, as it is in the



preferred embodiment, it is computationally efficient to use the final position computed for pulse N as the initial position for pulse N+1.

As mentioned above, before the force vectors of the various receivers are combined into a single vector they are each assigned a weight. Each force vector is not weighted evenly. Experimentation indicated that the receivers closer to the transmitter (located on the tracked object) tend to have more accurate TOF values. Therefore the first step of the weighting process is to multiply a receiver's force vector by the reciprocal of the receiver's TOF, consequently the weight given the force vector is inversely proportional to the TOF and to the distance from receiver to transmitter. The second step of the weighting function is to decrease the weight of a receiver based on the number of bad TOF values in the vicinity of the pulse in question. Bad TOF values are those which have been identified by various manual or automatic techniques as being invalid. For example, when the line-of-sight between the transmitter and a receiver is temporarily obstructed, the TOF value is actually for a path between the transmitter and receiver which includes one or more reflections off various surfaces in the environment. Plotting the TOF calculated by such an obstructed receiver over time when it is known that the vehicle (such as a camera sled) is moving slowly and steadily in relation to the sensor, usually reveals highly erratic TOF data. Depending on the severity of the fluctuations, such erratic data can either be entirely discounted or given very little weight. Once the quality of the TOF data is graded and weights assigned, the weights are first normalized and then multiplied by their respective force vectors.

As discussed above, the method employs iterative fractional application of force vectors to increase the reliability of the resulting positional data. Still, there is a risk that the iterations of the method may settle into a local minimum. This is more likely when the receivers are coplanar or nearly so, such that there is a local minimum on the wrong side of the plane. One way the method minimizes the probability of a local minimum is through the choice of the initial position. In the coplanar receivers case, the initial position is chosen to be much farther away from the plane than would be possible due to physical constraints of the environment and/or the range of the ultrasonic equipment. For example, if the receivers are all mounted near the ceiling of a room in which the transmitter is located, the preferable initial assumed position would be a position with the Z (vertical) coordinate well below the

floor of the room. As the computed position moves towards the plane by a fraction of the combined force vector it is more likely to settle into the actual position rather than some local minimum. Of course, after the first position, the algorithm uses the previous camera position as its assumed starting position.

5        Operation of the Positioning System. Assume the equipment is in place, has been initialized, calibrated and is ready for use. If film is being shot, an operator, while holding a smart slate 64 which has previously been jam-synched with the production time source 60 or 56 in view of camera 28, activates the smart slate 64. Smart slate 64, upon being activated, displays the clock time. The displayed time is thus captured in the picture as recorded on  
10 film and can later be associated with a particular frame or picture.

The Master SMPTE clock 60 or the VTR 56, used as the production time source, also sends the time over cable 98 or 96 respectively to time code translator 54 which converts the SMPTE time in the form of HOURS:MINUTES:SECONDS:FRAMES to an RS-232  
15 format for forwarding to camera tracking control computer 46 over cable 52. This time based data stream becomes the first of three serial inputs which the camera tracking control computer 46 receives. Meanwhile, camera tracking control computer 46 activates electronic compass 30 in camera sensor 24 by sending a command by way of the ultrasonic controller unit 22. This command initiates a continuous data stream from the electronic compass 30 which gives heading, roll, pitch, three axis (X, Y, Z) magnetic field information,  
20 temperature, and error code (as applicable) and a checksum.

This data stream, then, becomes the second of the three serial inputs received by camera tracking control computer 46. Camera tracking control computer 46 also sends a command to ultrasonic controller unit 22 to start an ultrasonic measuring cycle. Upon this command, ultrasonic controller unit 22 sends a signal to ultrasonic transmitter 32 which  
25 emits a burst of ultrasonic energy. The ultrasonic signal is received by ultrasonic receivers 36 which send signals back to ultrasonic controller unit 22 by means of cables 38. Ultrasonic controller unit 22 measures the time delay between the start of the triggering output pulse of ultrasonic energy and receipt of the leading edge of the return signal from each of ultrasonic receivers 36. Ultrasonic controller unit 22 then sends data to camera  
30 tracking computer 46 which, for each ultrasonic receiver 36, provides the time delay as calculated by ultrasonic controller unit 22. This input from ultrasonic controller unit 22,

becomes the third of the three serial inputs received by camera tracking control computer 46. Each of the three serial inputs is time-stamped as it is received by tracking control computer 46. These time stamps then become the key to allow camera tracking control computer 46 to correlate the above three serial input data streams with the SMPTE code.

5

### Recording Topography

To facilitate the map-based authoring and editing, it is helpful to record the positions of relevant features of the topography of the camera tracking area. In the preferred embodiment some of the features, particularly the paths, are placed in the map-based editor.

10 The paths can be drawn manually such as in a draw program or created from tracking data supplied by the tracking computer. A map displaying the drawn or otherwise recorded features is then generated by the system for use with the map-based editor.

The features recorded could include, the location of all corners, turns, intersections, and end-points of all paths in the environment and the location of all features with which a user may interact while playing the production. The location of these features, such as the corners or intersections of two or more paths being navigated, can be determined by any means known in the art. Further, such features can be recorded by use of a wand-type transducer according to the invention using, in the preferred ultrasonic embodiment, the following steps:

- 20
1. Activating a wand type ultrasonic transducer at the position of the feature whose location must be determined,
  2. Recording the time of flight from that point to a plurality of receivers placed at known locations,
  3. Calculating the location of said feature, and
  - 25 4. Plotting the topography of the environment.

The wand tracking data is generated using the same method as used to calculate the X, Y, and Z coordinates of the camera as described above. Alternatively, these positions can be measured by any other means known in the art and the topographical measurements recorded in the map-based editor.

30

Special Filming/Taping Camera Path Techniques

The most important additional task that should be done in the capturing step is to insure that there is adequate coverage by the camera of the environment being filmed or video taped. Besides recording the position and orientation of the camera for each frame of video, it is equally important that there be camera shots taken from sufficient headings along each path so that seemingly seamless transitions can be constructed as viewed by a player of the production while navigating the environment. Special shooting guidelines should be followed if these transitions are to look realistic.

Where a 90 degree field of view lens is being used only four shots are required to record a full 360 degree field of view. Simplistically, the shooting guidelines suggest that for each view, the scene should be shot, (for illustrative purposes only), traveling north to south (lens facing south), south to north (lens facing north), north to south (lens facing east), and north to south (lens facing west). Polar notations are for illustrative purposes only. Otherwise stated, for any given pathway, the camera should travel each path four times, each time with the nodal point of the camera lens directly on the path and the camera heading 90 degrees offset from the previous traverse. In this way four video records of each path are created: a "forward" facing record; a "rearward" facing record; a left-facing record; and a right facing record. As previously stated, it is very important that the nodal point of the camera lens travels down exactly the same path for each heading so that the resulting frames will line up properly for smooth transitions. The camera positioning system, when used in real time, can assist in accurate path tracking. Accurate path following results in frames of each orientation which share the same nodal point of the camera lens. By having such well-aligned camera shots looking to each quadrant from any given point, it is possible to link together like-looking frames from the intersections of these quadrant shots in order to seam together those frames as would be seen by a 360 degree, or lesser, pan from any given point.

Of course use of any other shooting technique available in the art would also fall within the invention. In any event, it is important that lighting and other scene elements remain constant during the capture of a given pathway.

While the 90 degree angular spacing appropriate for use of an anamorphic lens is preferred, any other workable combination of lens angles and camera headings could be

used. In order to provide 360 degree seamless transitions such lens angle and camera heading combinations, when combined together, must provide a full 360 degree coverage of the environment at any point.

While the seamless coverage of a full 360 degree view is preferred, it is possible to work with lens systems that capture less than the entire panorama and therefore leave seams in the captured view. These seams would only become visible during turns, and their presence can be made less noticeable by using transitions that distract the eye. For example, when an "accordion" transition is used, with two frames which are spatially related but not seamlessly connected, it is difficult to tell whether there was a seam or not. Consequently, spatial video productions can be made with video providing less than the preferred 360 degree coverage with acceptable results. Where there are seams or gaps in video coverage, transition types not requiring full coverage and edge-matched frames can be used. Many such transitions are well known in the prior art and will not be further discussed here.

#### Further Tasks

The capturing step is also where all of the various media elements (also known as media assets) called for in the script, including video, movie, graphic, animation, and audio segments are digitized, compressed, converted to an internal format, and stored in the computer. One particular type of asset described below is called a multiple-view object or M-view.

While taping, digitizing, compressing and storing the various spatially-related elements of the production, the six-axis tracking information is preserved. The capturing step also includes conversion and storage of the topography data.

Objects and M-Views. An important feature of the invention is the ability to augment a video scene with bitmaps of objects. Footage of a bare tabletop can, at run time, show a table with a vase on it, or a frog or any other bitmap image. Information about what object to draw, where to draw it, and how to draw it must be entered into the system during the edit step so it can be fed to the run time engine of the final production when needed.

This is accomplished according to the process of the present invention by a combination of logic steps and an innovative M-view data structure. The logic uses object oriented terminology. The physical item to be entered into the production is referred to as an

object. Particular depictions of the object are referred to as instances. For example, you may wish to place a vase on a table in a particular scene of a production. The concept of the vase is an object. Various instances of the "vase" object could be empty vase, vase with red flowers, vase with wilted flowers and broken vase. Each of these instances would have a different M-view associated with it, but would share many properties with the other instances in the object class. In the ensuing discussion, object classes are not discussed and the word "objects" may refer to an object class or an object instance, depending on the context. The relationship between object classes and instances is discussed in more detail below.

The invention includes a process for creation and insertion of objects into video that were not there when the video was created. Traditionally these objects have been called "sprites." Sprites are typically small bitmaps which can be overlaid and moved on other bitmaps. Sprites may be a single representation (one face, one size); but sometimes have multiple bitmaps associated with them representing multiple faces with one size; sometimes animated, one size; sometimes a few discrete sizes.

The process of the invention creates a new type of sprite, a multiple view object, called an M-view which has the ability to be inserted into the spatial video environment with the appropriate scale, angle and view for the place of insertion in the environment and the perspective of the viewer.

M-views, unlike prior art sprites, allow scaling to the desired size, rotating to appropriate face and moving to the right position. When inserted into video, M-views grow, shrink, rotate and move such that they appear to be part of the scene.

M-views may be created from a real object by capturing and digitizing images of the object from various angles or from computer models that may be rendered from various angles. There is no single preferred way as what is preferable will depend on the size, and complexity of the object as well as its importance to the production. There are numerous capture and rendering programs available which are capable of capturing or generating the required images. Regardless of the capture technique used, preferably at least one image is obtained for every 6 degrees of rotation about each axis around which the object is to be rotated. The angles need not be uniform or about any particular axis. The needs of a particular production will dictate which views are to be captured. Such needs include the scripted location and behavior of the object for which the M-view is being created. Such

considerations include, for example, whether the object is above or below the plane of the viewer, and whether the object rotates or moves in any other manner. In the productions created by inventors it has usually been sufficient to take pictures at uniform angles around the object's vertical axis. In the preferred embodiment the M-view will change size and orientation in response to the receptacle in which it is placed. However, shadows and perspective are not adjusted. The number of angles and the resolution at which the M-view is rendered is chosen by the editor. Choices are made as a function of the importance of the M-view in the scene, and the importance of high resolution or precise angle control. The importance varies depending on the shape, size and color of the M-view.

In the preferred embodiment, M-views are created as assets incorporated into the production at edit time, they are not created on-the-fly. Using a fast computer with a fast 3-D rendering algorithm some or all of the faces or resolutions stored in the M-view catalog of the preferred embodiment could be eliminated by storing the computer model of the M-view object and rendering the object on-the-fly at runtime. Caching or previous frame renderings might reduce the computer rendering required in such a system. Further, by rendering on-the-fly, the developer could control additional elements such as lighting, which could vary for the same object in different receptacles. Further, one could use all 6 camera position values to produce a correct image, rather than an approximation. Further, under certain conditions one could even add shadows from the lighting using an alpha channel. However, the computing requirements involved in such on-the-fly rendered objects are significant. In today's computers use of such on-the-fly rendered objects would dramatically reduce either the playability of the production or the number of M-views which could be used at one time and would therefore pose an undesirable reduction in the interactivity of the production. Thus, while use of such objects is within the scope of the invention, it is not a preferred embodiment at the present time.

When the bitmaps for the M-view views are created, it is important to create a transparent area around the object (unless the object is rectangular from all sides and always exactly fills the bitmap). This could be done with an alpha channel that tells the relative opacity or transparency of each pixel, or it can be done with a reserved color or color index that means the pixel is to be treated as transparent. In any case, the pixels at the edge of the object should not have anti-aliasing with the background color that is not part of the object or

the background color will be visible later, tingeing the edge color of the object. It is difficult to remove the blue screen color from the edge of real object.

Aliasing is an artifact well known to those skilled in the art which occurs due to limitations in the sampling frequency which reduce the clarity of the picture, creating a noticeable edge effect. Anti-aliasing is a way to make images smoother by choosing a pixel's color to be a proportional average of the colors that it might have been. Normally use of anti-aliasing algorithms will reduce edge effects; however when used with blue-screened overlays, the edges of the M-view remain tinged with the replacement color. Even the best techniques currently available often leave a blue tinge to the object's edge during filming of actual objects on a blue screen, and object edges are often tinged with blue even without use of anti-aliasing techniques. Therefore, where possible computer generated objects are the preferred objects for M-views. The preferred embodiment uses a reserved color technique where magenta is reserved for transparent parts of the bitmaps.

The bitmaps that are digitized from the real or modeled object may be kept in any convenient format; the preferred format is that provided in the Targa graphics file format defined by Truevision, Inc. of Indianapolis, IN. The Targa format was chosen because of its compatibility with all of the various post-processing steps applied for smoothing, sizing, cropping, and the like, and in particular compatible with the preferred SoftImage rendering software and PBM file conversion utilities used. The SoftImage software is published by Microsoft Corporation of Redmond, Washington. PBM utilities are published by Jef Poskanzer Shareware, available from Jef Poskanzer at [jef@well.sf.ca.us](mailto:jef@well.sf.ca.us) apple!well!jef and are generally available to those skilled in the art. In the preferred embodiment, the M-view compiler reads all of the bitmaps that belong to a single object and binds them into a single file for the benefit of the runtime system. Also in order to save storage space it is recommended that a suitable compression technique, such as run length encoding, be used to compress the images. Further, the preferred embodiment of the editing system allows the user of the editing system to chose the resolution at which each M-view is to be retained.

The preferred compression technique uses run length encoding using the principles set out in Foley & van Dam.

The steps of the preferred method of creating an M-view are shown in the process flow diagram of **FIG. 9**. When creating an M-view of the preferred, computer-rendered



object, the first step is often to build a 3-D model of the object, preferably using a computer workstation, such as sold by Silicon Graphics, Inc., programmed with a computer aided design program such as SoftImage from Microsoft Corporation. After the model is built, its various views are rendered using the same program and the bitmap files representing the various rendered views are stored in the Targa .TGA format. The number of faces captured and the direction or directions of rotations will depend on the use of the object in the production. Generally, the capture of faces around the axis of rotation at 6 degree increments is sufficient. The resulting 60 bitmaps representing the 60 faces stored in .TGA files are then moved from the rendering system to the PC. The files can be further manipulated using the preferred PBM Utilities or similar graphics utilities.

Once the various angle pictures of an M-view are in acceptable condition they are imported into the spatial video edit system using an M-view compiler program which takes the 60 (or however many) designated faces of each M-view and loads them in a data structure that can be loaded from a disk file to memory and quickly accessed to get any desired face. At edit time this same structure is used for accessing the faces for placement on frames of video. This directory structure is shown in **FIG. 10**. Virtually the same data structure is used in the edit system as well as at runtime for performance reasons. The access mechanism uses map-based data to determine what view to overlay on the display. The M-views themselves remain in the catalog data structure, but are represented by references in the edit time database which ties M-views to specific video clips and frames. At runtime, a reduced version of the M-view catalog database structure is copied into the runtime data hunk (see below). Unused M-views are not copied over.

Referring again to **FIG. 10**, the editor accesses the M-view directory as being under the catalog directory. Other subdirectories include the Album Directory which is also shown. The edit system user, using the process of **FIG. 9** can decide on the resolution and angles which will be a part of the final runtime production.

At runtime the M-views relevant to a navigable area are preferably kept in memory, therefore, there is a trade off in quality of realism versus memory consumption on number of views and resolution of the M-view image. The goal is to keep the amount of data in memory as small as possible to save memory resources for other parts of the system. At the same time, M-view resolution must be high enough to satisfy the production's visual

requirements. The resolution and number of faces for an M-view can be chosen on an individual M-view basis so that more important M-views can have better resolution.

In another preferred embodiment, low resolution M-views are kept in memory to allow for rapid navigation of the environment, however, if a user pauses to explore a particular area, the M-views in that area are updated with high resolution M-views from the storage medium, thus allowing for rapid navigation and higher quality reproduction without taxing the memory system where the user might stop to examine an M-view more closely.

At edit time the editor examines the available M-view angle faces at certain points along the camera path and designates the desired M-view angle. For example, the editor can designate a starting angle for the first frame in which the M-view appears, one intermediate angle near the midpoint of the M-view's passage through the frame sequence, and ending angle for the last frame and allow the system to interpolate the angles for the intermediate frames. The interpolation used is simple linear interpolation between the user-selected angles.

Further, where the mathematically correct face, as calculated by the preferred linear interpolation technique, is not available, the M-view access program chooses the nearest available face to the requested one. This means, for example, that during editing the user of the edit system can assign an angle of 10 degrees at the beginning of a path and 50 degrees at the last frame on the clip where the M-view is visible. The total number of frames in which the M-view is visible might be 1000 frames. In these 1000 frames the angle of the M-view will adjust between 10 degrees and 50 degrees in increments chosen by means of linear interpolation. The M-view that was available at edit time may have had 60 faces, so the angle of 10 degrees was represented by the 2nd face (12 degrees) and the angle of 50 by the 8th face (48 degrees). At a later edit stage the editor may have decided to reduce the storage requirements of the M-view by changing the number of faces from 60 to 12, i.e. every 30 degrees rather than every 6 degrees. Once this decision is made and the M-view structure reduced accordingly, the face at the beginning of the clip will be represented by the 12th face (corresponding to 360 degrees which is the same as 0 degrees which is as close to 10 as we get with only 12 faces) and the angle of 50 degrees is represented by the 2nd face (60 degrees). The first face will be used in the middle of the clip as well (30 degrees). In any case, the system chooses the nearest available angle to represent the requested angle and the

system adjusts automatically to the developer's reduction of the number of available angles (faces) to save memory or to the developer's increasing the number of faces to improve the angular resolution.

M-views have associated with them not only angles, but also a size. In the preferred embodiment an M-view is associated with a receptacle which is constructed during the editing process discussed below. A receptacle has associated with it a size and a position. When the M-view is in a receptacle, the receptacle size and the M-view sizes are combined by multiplying the two sizes together. Thus, as the receptacle size increases, the M-view size also increases proportionately.

It may be inaccurate to speak of an M-view being "in" a receptacle. The M-view gets its size and origin from the receptacle, but, depending on the size of the M-view, the M-view bitmap could be much larger than the receptacle dimensions. The receptacle can be sized to any size using the receptacle edit tool in the spatial video edit system. Its initial size is set by the developer depending on the use for which the receptacle is intended and the environmental feature with which the receptacle is associated. It is often useful to scale all receptacles in a production the same. That is, make all receptacles the size they should be to contain a figure or feature of known size. For example, all receptacles could be scaled to contain a one-foot object. In this way, the size of M-views relative to the video environment will remain consistent no matter in which receptacle they are placed.

As the size of the M-view face actually displayed is determined by multiplying the M-view size number by the receptacle size, the M-view size is a function of the size of the receptacle. The M-view bitmap is scaled according to the multiplication to the proper size. As it is important in a production to create receptacles of a constant size within the video environment, it is also important to provide various M-views with size numbers which reflect the relative sizes of the object in the M-view. For example, where the receptacle is sized to reflect a one-foot square, the M-view of an object which is one foot in height should have a size number of 1. An M-view object which is three feet in height should have an M-view number of 3. Finally, an M-view object which is half a foot in height should have a size number of 0.5. Therefore not only will the M-view grow or shrink as the receptacle enlarges or becomes smaller, but also an M-view with a small size will appear smaller in a given receptacle than an M-view with a larger size and both will be properly scaled for the

environment in which they are depicted. This allows different sized M-view objects which are placed into the same receptacle to keep their relative size and to appear true to their absolute size in the video environment.

When the M-views are overlaid on the video, either at edit time for placement or at runtime, the system must be able to both translate (position) and scale (size) the bitmaps as well as render just the non-transparent pixels to the screen. The M-view software implements all three operations in a single step for efficiency, although they could be done one at a time on fast enough hardware. Run length encoding compression and decompression used for the M-view faces (bitmaps) lends itself well to rapid decompression, scaling, and transparency; however almost any bitmap compression/decompression format could be used as long as decompression is fast enough and memory requirements are small enough to fit with the resources devoted to M-views in a particular production.

A number of techniques are available for updating an M-view or other type of sprite animation on screen; typically a double or triple buffer is used. In a double buffer technique one buffer is under construction while the other buffer provides a stable image for the display hardware. The double buffer technique is preferable to writing directly to the screen because it provides a more stable image for display. In the preferred double buffer technique, the scene under construction is completely constructed in memory rather than in the screen buffer, then the completed image is copied in its entirety into the display buffer. Where the production will be played on machines with video display hardware containing sufficient video memory to allow for the double buffers to be built in video memory and supporting video page swapping, use of video memory storage and video page swapping obviates the need to construct the scene in memory and then copy to the display buffer.

In the preferred M-view system embodiment, every new frame completely replaces the previous frame because of the way the preferred SoftMotion player works. As soon as the MPEG is decoded into a display buffer the M-view system overlays it with one or more M-views. The contents of the display buffer are then sent to the display.

M-views may use animation as well as static bit-maps. This can be done by choosing bitmaps that reflect the changes in position and posture of the object in the M-view.

M-views and receptacles may also have a logic flow associated with them. For example, a flow program may allow a receptacle to accept or reject different objects or allow different objects to operate on each other.

Picking up an M-view. The system provides the ability for the user to “pick up” an object represented by an M-view by creating a special cursor. When the user picks up an M-view from the video, the system stops overlaying the M-view on the video so it appears to be removed. The mouse cursor changes to a representation of the object picked up to give the user feedback that the user is now carrying the object. The cursor/object can be dragged across the screen off of the video portion and, for example, into a container shown on another portion of the display.

Visuals. The term “visual” is used in the preferred embodiment of the system to refer generally to the preferably rectangular areas of the display in which data in the form of bitmaps (still pictures), text block, label, video and the like are displayed. Of course, while the preferred shape is rectangular, any other shape could be used, although at the cost of use of significantly greater processing resources. Visuals are the primary building blocks for building the spatial video production display. The visuals in which spatial video or dramatic video is played are called “viewports.” Visuals which frame bitmaps are called “pictures.” User written embedded, external programs, preferably OCX programs, are displayed in windows called “custom.” OCX’s are further discussed below in the section on embedded external programs. All other visuals are called “windows,” including the window in which the entire production resides. Thus a display of a spatial video production made according to the invention would have a window which fills the entire display screen, centrally located within this window might be a large viewport in which the spatial video is playing. Above, below and/or to either or both sides of the viewport a window could be located which could contain inventory, a dialog window, or similar constructions useful for interacting with the program. Further, any such window might overlap one or more other windows at certain times during the production. For example, in a production of a museum, clicking on a hotspot covering a statue might cause a flow which in turn displays a picture showing a large bitmap of the statue.

Visuals can be placed one within another in a “parent-child” relationship. For example, the entire spatial video production preferably plays within a window. This window

contains various "child" visuals, such as pictures in which bitmaps are displayed, surrounding the viewport in which the spatial video runs. The various bitmaps in the visuals surrounding the video window may react to events within the video window by altering contents in response to events occurring within the viewport. A custom window might display a user created OCX which, for example, may display a map visual of the video environment on which a red dot appears representing the position of the player within the environment. The OCX could take the camera location data from the runtime video interpreter (see below) and, as the player moves, cause the red dot in the visual to change its location on the map visual to reflect the change in position. A picture visual of a button might have a flow associated with it which, when the button is clicked on, causes the bitmap showing a red button to be replaced with a bitmap showing a green button. The bitmap pictures which may be placed in visuals are also preferably stored in the catalog structure in the album directory, as shown in **FIG. 10**. The treatment of visuals in a preferred embodiment of the system is further illustrated in the database documentation contained in Appendix B and the Flow Language documentation contained in Appendix A.

The connection of visuals and other assets to the spatial video production through the use of the database and the spatial video edit system is a unique aspect of the system as is the fact that the viewport is the visual in which the spatial video of the present invention runs. However the concept and mechanics used for displaying visuals by the system is not unique.

Spatial Video System Database. Every production made with a preferred embodiment of the spatial video production system has two major components: media and data. The media component includes video, audio, bitmaps, cursors, etc. The data component is the complex set of pointers and characteristics, logic flows and other items that combines the media assets into a production. Every hotspot, for example, may have position information (X, Y, width, height) for every video frame in which it appears. Another example of data in the productions is data items relating to connectivity between various clips in the production: you can turn right at frame 947 in video clip 238, and if you do, you will end up at frame 35 in video clip 67. This right turn should be accomplished using a "push right" transition. Many other kinds of data are contained in the production. The spatial video edit system is simply a software tool that allows users to enter, review, and tune the data in the production.

Data fields used in the spatial video database cover all the "standard" types (e.g., boolean values, numbers, strings). More exotic data types like memo and large binary object are not used in the system database. In this respect, the spatial video edit system is very undemanding as almost any available database technology supports the data types used in the invention.

Since productions may be very large, in the preferred embodiment the spatial video edit system permits multiple members of the development team to be editing the data simultaneously. Changes made by one editor are instantly visible to all other editors. This multi-user simultaneous access is an important aspect of the preferred embodiment of the spatial video edit system. While the spatial video edit process does not require custom database technology, it preferably requires a database technology which supports multiple user simultaneous access.

Data in spatial video productions is heavily inter-related. For example, a video asset may contain frames which have been designated by the edit system as a video clip associated with a clip group associated with a path. The frame may have a hotspot associated with a defined area of the frame. The hotspot, when clicked on, may run a flow (user-specified game logic) that activates a receptacle located elsewhere on the frame and inserts an M-view into that receptacle. All of these entities (video asset, video clip, hotspot, flow, M-view, and receptacle) are described in records in the spatial video database.

As should be apparent from the above example, each entity description in the spatial video database refers to other entities. Because the database contains many references from one entity to another, it is important that a consistent reference form is established in the database structure. For example, the database description of a ROOM will identify the MAP in which the ROOM appears. Similarly a CLIP will list the ROOMs that appear in the CLIP.

In the preferred form names of specific entities are associated with unique id numbers contained in internal database. Every time an entity is created it can be entered into a master registry and assigned an id. The id can be an arbitrary number with enough bits that its uniqueness can be assured. Preferably the id is a 32 bit number.

The preferred database is a relational database, which must deal with references which are lists. Sometimes a reference from one database record to another is unique in the sense that only one item is referenced per record. For other references, such as OBJECTs in

the production, there may be a list of properties which the object uses. The preferred approach to this problem is to define a separate table that records which objects have which properties. Each record in this set lists exactly one OBJECT and exactly one PROPERTY. There may be many records with the same value for the OBJECT field and many records with the same value in the PROPERTY field. However, for each combination of OBJECT and PROPERTY, there is at most one record.

The frequent occurrence of references in spatial video data increases the importance of avoiding database corruption. When a single record is corrupted or accidentally deleted, not only is that record lost, but in addition all the records that refer to it become, in a sense, corrupted because they now refer to a record that no longer exist or is damaged. Therefore another factor in choosing a database which can be used in the method of the invention is one which is robust and easily repaired.

The heavy usage of references to other records in spatial video data makes it desirable for the spatial video developer to have an interactive tool that will allow quick display of the values in any database record. When tracking a logic problem that generated a bad records, it may be necessary for the developer to see the data values in several records related to the bad one.

The above requirements can be met by a database engine that comes with an interactive data viewing tool, and also allows one to generate programs that access the engine directly. In particular, the preferred database engine is Microsoft's Jet<sup>TM</sup> engine, which supports data viewing through Microsoft's Access<sup>TM</sup> program. Jet has sophisticated features to ensure the integrity of the data, and to repair damaged databases. It allows efficient access from Visual Basic. And the Access program can be used to examine and repair individual values in the database.

While any number of databases and organization systems can be used to create the spatial video edit system of the invention, a typical edit database record format list is shown in Appendix B which is incorporated herein by reference. As can be seen in Appendix B, the database edit information is collected in logically organized tables which can be worked upon by multiple users simultaneously. In Appendix B several fields are described as lists. Although this is the right logical structure, the lists are actually implemented as separate tables as described above.



Relationship of Edit System Data to Runtime Data. The spatial video edit system database must store the data associated with the production being edited in a form that facilitates review and alteration. For example, the hotspot locations are displayed for review and adjustment by drawing hotspot rectangles on top of video frames -- not just by listing the current values in the hotspot data as numbers. The requirement that edit-time data be stored in a system which facilitates review and adjustment is at odds with the runtime requirements placed on the same data once it is part of the bound production. At runtime the production's data must be compact and accessed with efficiency. For this reason, even though the data accessed through the spatial video editor is essentially the same data that will drive the production in its final form, the format of the data is vastly different. The runtime data structure will be described in more detail below in the section on the production binder. The production binder is the system which, among other things, creates a finished production from the assets, edit-time database and other edit-time structures.

### Edit Step

The editing step includes means for combining and managing assets captured during the capture step for incorporation into a spatial video production. The edit system is the basic means of creating database entries which relate the various assets to the script and to each other in a spatial video production. The incorporation task is performed by the spatial video editor. As shown in **FIG. 11**, the inputs to the editor **14** are the spatial multimedia script **18** and the captured assets **100**. The output from the editor is a database **102** describing all of the relationships and behavior of the assets used in the production. This output and the media assets **100** are used by the binder **104** to create a final production.

The editor **14** and the database it creates **102** provide the edit system user the ability to define clip exits, transitions, roundabouts and the like, as well as user interaction mechanisms such as buttons, hotspots, receptacles, menus, and animated cursors, to create logic flows and to assign these flows to various events. Events, as used herein refers to a variety of synchronous and asynchronous aspects of the production.

Mouse events include the standard asynchronous interactions the user has with the system by means of the mouse where the production developer has chosen to allow such actions to cause the system to respond. For example, a mouse event in which the cursor

controlled by the mouse moves over an active hotspot or button or the like is such an event. Further, a mouse click while the cursor is over an active hotspot may be another such "mouse event."

5 The invention of spatial video as disclosed herein also introduces a new class of events sometimes referred to herein as "frame events." Such events are various actions programmed into the production which are associated with a particular video frame. An example of such an event would be the definition of a hotspot or receptacle in a particular video frame. If a transition has been defined for a particular frame, a transition event would be another frame event associated with that frame. Similarly, a transition event should be  
10 paired with a clip exit event which would be another frame event. Various types of events are discussed throughout this document. Creation of frame events is discussed in the sections of this document dealing with the edit system and the binder. Interpretation and execution of frame events are further discussed in the section of this document dealing with the runtime system.

15 A primary task of the edit system is to create an intuitive interface whereby the developer can construct the various database entries which associate the assets and logic flows in a meaningful way. While there are many tasks which the edit system and database perform which are not unique to spatial video productions and many aspects of such productions which are not unique, one unique aspect of the system is the idea of map-based  
20 editing where the map is actually a method of visualizing camera paths through the environment. As will be discussed in more detail below, one of the first tasks of the developer is to lay out camera paths on the map (called clips) and to associate video with these clips. This is done through use of the database to identify which video stream is associated with which map clip and which frame on the video stream corresponds to the  
25 beginning point of the clip and which frame corresponds to the end point.

Once the video assets are associated with the clip, the map actually becomes an association of video frames to two dimensional space. When editing is then done by selecting map positions on the map editor, it is actually being done by selecting frame numbers in the associated video clips. As will be seen in the following discussion, this  
30 revolutionary concept allows for many efficiencies in editing. It also gives rise to several interesting associations. First, the concept of the beginning and ending of a map associated

video clip (the MarkIn and MarkOut points discussed below) are end-of-clip exits. Similarly, intersections between two paths allow for transitions between a clip associated with one path (e.g. the north-south path) and a clip associated with another path (e.g. the east-west path). The point of such intersection shown on the map represents collectively the frames of each clip in the intersecting paths which were shot from the same physical camera nodal position. The editing of an intersection is actually the definition of a clip exit from one video clip to another at a specifically defined frame.

Similarly, a roundabout, where the player rotates to a different heading while maintaining the same position in three dimensional space, is actually a series of clip exits from one clip in a group of clips shot along the same path to another clip in the group which other clip was shot along the same path with a different camera heading. As such roundabouts are also frame specific as well as clip specific, they are defined in the database as designated "FROM" and "TO" frames on the specific clips. Once defined the associated clip exit can be used as a clip entrance as well if the player chooses to make the reciprocal turn or roundabout from the associated clip.

Thus, as used in this disclosure, the concept of clip exit is a record in the database used to define the frame in the FROM and TO clips where the user can execute a transition from one clip to the other as well as the type transition associated therewith. As previously discussed in this document and as discussed in the copending Special Effects Applications, common transitions include left turns and right turns which in the context of transitions from a clip in one clip group to a clip in an intersecting clip group are called intersection transitions and in the context of a transition from one clip in a group to another clip in that same group are called roundabout transitions. Clip exits can also be associated with other types of transitions. For instance, a clip exit can be defined on a frame where the next scene to be viewed is from the same clip or from a different clip but at a point not spatially associated with the FROM clip. These are called "teleport" transitions and could occur when the player solves a puzzle or is otherwise allowed by the program logic to enter into a different environment. Other exits could be to a dramatic video sequence or some such event which could be called by the system's flow logic. These exits are called flow exits.

Functionality of Edit Process

The process of editing a spatial video production involves the nine steps shown in **FIG. 12:**

1. Identify Assets
- 5 2. Organize Video Clips on a Map
3. Construct M-view Catalogs
4. Edit Intersection of Video Clips on the Map
5. Edit Roundabout points on the Map
6. Place Hotspots on Video
- 10 7. Place M-views and Receptacles on video
8. Edit the Logic for Hotspots and Flow Exits
9. Design the Screen layout for the production

The order in which editing is done is not rigid. In fact, one innovative feature of the editor is that multiple users can access a single production simultaneously to work on a single production. Thus many, if not all, of the steps of the editing process can be performed in any order. The steps of the edit process are discussed below.

Identify Assets. Assets are any data (files) in a production which are generated outside of the edit system. These files remain outside of the Edit database because many of them can be quite large. The types of assets which the edit process typically utilizes are shown in **FIG. 26:**

- a. Video Assets - Files which contain digital video, with and without audio data, such as spatial and dramatic MPEG video streams, MPEG audio streams and MPEG system streams.
- b. Image Assets - Files which contain single images (bitmaps or pictures) for use in constructing the visible user interface
- 25 c. Objects: M-view Catalog Assets - Files which contain multiple bitmaps representing Multiple views of an object.
- d. Audio Assets - Files which contain MIDI data or files which contain digitized WAV or other digitized audio data.

- c. Position Data - Files which contain camera tracking coordinates for a video clip.
- f. Cursors - Files which contain single cursor images or animated cursor images.

In order for the edit system to construct the spatial video environment, the edit system must know information about the assets. This is accomplished through the Asset Identification Process by which location and characteristics of the various assets are entered into the edit system. Such associations can be carried out by any method known in the art. The method of a preferred embodiment uses a Microsoft Windows compatible graphical user interface. The interface characteristics conform to the Windows conventions set out in the *Microsoft Developer Network Development Platform: SDKs and Operating Systems*, January, 1996, published by and available from Microsoft Corporation. The various menus used in the preferred embodiment of the edit system are attached hereto as Attachment 1, and incorporated herein by reference. The concepts and functions set out below are independent of the Microsoft paradigm and could be adapted to any operating system.

In the preferred Microsoft Windows compatible interface, the Edit system asset management is accomplished using the library window. The library window contains a "tab" control with one tab for each of the asset types which are utilized in the editor. The screen representing this window is shown on the fifth sheet of Attachment 1. When each tab is selected the Library window displays a listbox of all the assets of that type. The listbox contains a representative thumbnail for the asset, The name of the asset and some of the significant properties of the asset. The representative thumbnail may be symbolic (e.g., A representation of a sine wave for digital audio) or it may be a reduced thumbnail image extracted from the asset (e.g., The first frame of a video file). This combination of tabs for categories, listbox with thumbnail pictures, and display of relevant information allows for quick recognition and selection of assets when they are needed in the production.

The asset identification process consists of three steps: user's identification of an asset file through the system's file dialog window; system's examination of the file to determine its attributes; and system's update of the database with records identifying the asset selected by user.

User begins the process by identifying the file sought through the system file dialog window. The user activates the editor's main window menu item for files which in the

preferred windows environment is titled "File," and clicks on the "Import Media" command which is then displayed. See Attachment 1, sheet 1. Clicking on import media displays a system File Open dialog which allows the user to select a file. In the next step the edit system examines the selected file to obtain its attributes. Based on the type of the file,  
5 different relevant information is gathered.

For Video, the relevant attributes are:

Number of frames

Frame size (Horizontal and vertical resolution)

Playback frame rate

10 For Bitmaps, the relevant attributes are:

The Horizontal and vertical resolution

The number of bits of color depth

For MIDI data, the relevant attributes are:

Playing time in minutes and seconds

15 For WAV data, the relevant attributes are:

Playing time in milliseconds

Sample size (4, 8, 16 bit)

Sample Frequency

For Cursors, the relevant attributes are:

20 The number of frames (if the cursor is animated)

For Camera tracking data,

The frame rate of the data

The SMPTE time code of the source video to which this data corresponds.

25 As the third step of the asset identification process, the edit system updates the database (described below) with new records to identify the assets. These new records are created to record the data selected by the user. An ID is allocated for the asset and the file location is stored in the Library table. Other relevant information is stored in the respective asset table. All references to these assets within the software is through the ID.

In the case of Camera tracking data, it is necessary to read the camera position records and store them as records in the Frame position tables. This involves creating a record in the Frame position table for each ASCII record in the camera track file and copying the values into the table. See text accompanying **FIG. 7**.

5        Organize Video Clips on a Map. A central task in creating a spatial video production is to organize portions of the motion video captured by the camera which portions reflect the movement of the camera along a natural path in the environment. These path associated video segments are called clips. In every spatial video clip the camera moves along a path. It is a useful aspect of the Edit system for the system user to be able to see these paths  
10       plotted on a map. **FIG. 13** shows a portion of such a map showing walls **126** as solid lines and a camera path **124** as a dotted line. Pages 6 and 7 of Attachment 1 show the preferred interface for displaying and drawing a map. Once the clips are associated with the camera paths of the map, the map display makes it easy to locate a desired clip. For example, if you want the clip where the camera went down the middle of the main hall **124** of the castle,  
15       click on the path drawn down the main hall of the castle. The display also makes it easy to find desired frames in a clip. To see the frame where the camera passed the suit of armor, click on the camera path where it comes nearest to the icon representing the suit of armor receptacle.

The map display also makes it easy to identify path intersections. How to identify  
20       path intersections will be discussed in more detail subsequently. When viewing the Edit system map, the user may zoom in or out, and pan (scroll) horizontally and vertically, and rotate the map as desired. Zooming allows one to see more or less detail in the map. Panning shifts the view to a different portion of the map. Rotation could be used to make the top of the map correspond to a direction other than north. Straight lines may be drawn on the  
25       map to represent walls, furniture, or other objects for purposes of giving context to the camera paths.

One aspect of the map approach to spatial video is that not all clips have unique paths, and therefore cannot be seen as individual clips on the map. This condition is usually the result of the camera being run over the same path, to record the topography from  
30       different headings, e.g., forward, left facing, right facing, backwards. The map editor deals with this characteristic of the technology by introducing the concept of clip groups. A clip

group typically contains all the clips that share the same path. In the preferred embodiment, the group is represented in the map by a path of double thickness, as opposed to the thin line representing the single camera path. The user of the map editor may click on a double thick path representing a clip group and then select any of the individual clips in the group.

5       The edit system uses a map tool which is a draw-like tool, driven both by user input and by the camera positional data in the system database. The functions of zooming, panning and rotation are standard, as is the ability to draw and display the wall segments (straight lines). The map tool creates a two dimensional representation of the environment wherein lines represent the path of the camera through space as well as other significant features of the environment. The edit system map tool allows the user to draw camera paths on the map. This is useful for planning shots before any actual camera position data is available. It also supports the required functionality for these planning sketches to be used through to the completion of the production even when camera position data is not available.

10       All geographic coordinates in the map are stored in units or millimeters measured from an arbitrary origin.

15       The drawing of camera paths is straight forward in those cases where the database has been populated with frame position data for every video frame. In such circumstances the primary task is to normalize coordinate systems so that the coordinates of the frame position data place the clips on the map at the proper map coordinates. When the camera paths are drawn instead of derived from positional data, the map tool allows path drawing in two formats: a series of smooth curves or a series of straight lines. In the case of straight lines, the line segments are joined end to end and the corners where they meet may be hard points or rounded into smooth curves, with the degree of rounding controlled by the user.

20       For each path on the map, when the user clicks on a point the edit system can determine which camera path the click is on; when the user clicks on a path, the system can determine which frame number of the clip (or frame numbers of the frames in each clip of the group of clips represented by the path) was captured at the location of the click; and, given a clip and frame number, the system can find the pixel in the map display corresponding to this clip and frame number.

25       In the preferred embodiment of the map editor, clips are associated with camera paths by drawing the clip paths on the map (or associating camera tracking data) and associating



the appropriate video asset with the clip and identifying its MarkIn and MarkOut points. These MarkIn and MarkOut points are recorded in the database.

The steps for laying out clips in the map editor is illustrated in **FIG. 14**. It is important to understand closely related concepts relating to video productions. Dramatic video is normal, lip-synched non-navigable, linear video. Video assets as usually used herein refer both to dramatic video assets, which may also be included in a spatial video production, and to spatial video assets. Spatial video assets may or may not be associated with a camera path on a spatial video map. A spatial video asset is incorporated into a production when it, or a portion of it, is associated with a "clip." A "video clip" is a logical excerpt from a spatial video asset which has been associated with a map clip. A clip or map clip is a camera path line on the map of the invention with which a spatial video asset is to be associated. A clip group is a group of clips which share the same camera path location on the map, but have different camera headings, such as a forward facing clip, a rearward facing clip, a left facing clip and a right facing clip. Consequently, in the edit system a clip may begin as a camera path line on a spatial video map. At the time it is created it usually incomplete. Clips have a property which requires the association of a spatial video asset. Thus, a completed clip will always have a spatial video asset associated with it.

STEP ONE: Create a Map. The clip layout process is iterative with the various paths being laid out on the map and the video clips being associated with their respective paths. To initiate the process in the preferred Microsoft Windows 95 environment of the graphical user interface, the user creates a Map by clicking on the New Map command button on the Main menu and toolbar. In Attachment 1, the new map button is the second button from the left on the primary screen and contains a symbol resembling a compass rose. This results in the allocation of a new ID and the creation of a new record in the MAP Table of the database. The Map Editor tool, see Attachment 1, page 7, is then displayed and initialized to edit the video clips for the newly created Map record.

STEP TWO: Draw the first clip in a group. As shown in the second step illustrated in **FIG. 14**, the user then creates a new clip by clicking on the New Clip command button on the Main menu and toolbar of Map Editor, Attachment 1, page 17. This results in the allocation of a new ID for the new clip and the create of new records in the VideoClip and SpatialClip tables of the edit database. The map must be initialized with an empty Frame

position record to start drawing the camera path so a new record for the video clip is added to the Frame Positions Table with null values (0) for X, Y, and Z and passed to the map tool.

The map tool is then set in the "draw new clip" edit mode. The user clicks on the points to define the camera path and right mouse clicks to terminate drawing. All smooth  
5 curves used for the camera path sketches are preferably Catmull-Rom splines. Curves drawn with a drawing tool which generates Catmull-Rom splines are preferred because use of Catmull-Rom splines result in:

1. "Smooth" curves (continuous curves with continuous derivatives)
2. Curves that run through a set of points with no limit on the maximum number  
10 of points
3. Curves that run exactly through the points (not just comes close to them)
4. Curves where the points along the curve can be generated at any scale, using simple multiplication and addition operations (enabling fast code)

The third point above is particularly important for usability. Most spline algorithms,  
15 when given a set of control points, will draw a smooth curve that is guided by the control points, but does not pass through them. It usually requires trial and error to position the control points so that the curve will go where the user wants it to. Catmull-Rom splines, on the other hand, pass directly through all given control points.

When two straight line segments meet at a corner, the corner can be rounded to  
20 varying degrees as follows. First map the degree of rounding desired into a fraction between 0 and 0.5. Zero means no rounding; one half means maximum rounding. Call this fraction R. As the two line segments converge, erase the part of each that is closest to the intersection. R determines the fraction of each line to erase. Fill in the resulting gap with a Catmull-Rom spline. Each such spline requires four control points. Use the four endpoints  
25 of the (partially erased) line segments. Note that this technique is used to generate curves within a continuous line at places other than line intersections. As discussed above in the Capture Step: Optical Enhancement section of this document, when using the preferred anamorphic lens it is preferred that intersections of two different camera paths occur at 90 degree angles.

30 Further, spatial video allows the capture of natural paths regardless of the shape of the resulting line. However, it is very useful when the capture technology requires multiple

camera passes on the line to create the various path views, that the path be one which is easy for the camera to repeatedly and accurately traverse. The smooth curves generated by use of the Catmull-Rom spline drawing tool are such curves. Thus use of paths laid out with such a drawing tool can enhance the accuracy of the camera path capture process.

5       Once the sketched curves have been entered, their lengths can be computed by generating a dense series of points along the curves and summing the lengths of the small line segments that would join all the points. This gives us the total length of the sketched camera path in pixels. Assuming the camera is traveling at a constant velocity during filming, this also allows one to find the point that is any fractional distance along the curve.

10      For example, one could find the point that is exactly half way from the beginning to the end of the curve. Or the point that is 9/10 of the way. This allows us to associate points along the curve with frames in the clip. If the clip has 1000 frames, then frame 500 may be found at the point exactly half way along the curve. Simply generate the same dense set of points along the curve and total the lengths of the tiny line segments until the running total is half of  
15      the known total for the entire path.

For those paths on which the camera traveled at constant speed, this smooth interpolation of frame numbers along the curve is sufficient. When the camera speed was not constant, spatial video edit system gives the user more control by allowing him to specify exact frame numbers at each control point of the sketched curve. (The control points are the  
20      points clicked to draw the curve in the first place). At each control point, the spatial video edit system user may specify two frame numbers: the FRAME IN and the FRAME OUT. The former specifies when the camera arrived at the point; the latter when the camera left the point. For any control point, these two values are frequently the same. They would differ if the camera stopped at the point for a while before continuing along the path, or if the point  
25      has been rounded into a small spline as explained above.

After each clip in a group of clips is drawn the video asset to be associated with the clip is located and edited if it has been captured and is available. This process is described in detail below in the text accompanying **FIG. 15**. However, clips can also be drawn during the conception part of the process prior to the capture of video. In such a situation, the video can  
30      be associated at a later time.

STEP THREE: Generate the Inverse Clip. After the first clip is associated, an inverse clip can be generated by duplicating the first clip and then changing its properties. An inverse clip is a clip which shares the same definition points as another clip but in reverse order. For example if clip1 starts at point A and goes to B facing point B, the inverse of clip1 starts at point B and goes to point A facing point A. To create an inverse clip it is necessary to first select the desired forward clip. In the preferred embodiment, to select a clip on the map one performs a left-mouse-click on the line which represents the clip. The edit system responds to a successful clip selection by positioning a small red block which represents the camera position on the path. Since the Edit System only displays one clip in a group at a time, alternate members of the group may be selected by selecting the appropriate member from those displayed in the right-mouse-click pop-up menu. The currently active (displayed) member of a group is stored in the clipgroup table and is indicated by a check mark adjacent to the list of member clips in the right-mouse-click pop-up menu. See Attachment 1, page 18.

After the appropriate forward clip is selected by clicking on its map location, click "Create Inverse Clip" command on the selected clip's right mouse button pop-up menu. If the Selected clip is not in a group already, this command allocates a groupID and adds a new record to the clip group table. The original forward clip is then assigned as a member of the group. A new record is added to the VideoClip table and SpatialClip table for the inverse clip, new FramePosition records are created and added to Frame Position table (based on the inversion of the frame positions for the forward clip). And the New Inverse clip is set as the active (displayed) clip.

As in the case for the forward clip, a video clip must be associated with the inverse clip. This process is described in detail below. Of course the video clip selected should be the one which corresponds to the camera traveling opposite the forward direction. Again, the appropriate (backward facing) video clip should be associated with the inverse clip and edited.

STEP FOUR: Generate Left Facing Clip. The left facing clip can also be generated by duplicating an existing clip in the group of clips and then changing its properties. In order to do so one selects an appropriate member of the group (which has the same direction as the Left facing video) from the list of clip members in the right mouse button pop-up

menu and then select "Create Duplicate Clip" from the right mouse button pop-up menu. If the Selected clip is not in a group already, this command allocates a groupID and adds a new record to the Clip Group Table. The original forward clip is then assigned as a member of the group. A new record is added to the Video Clip and Spatial Clip Table for the duplicated clip, new FramePosition records are created and added to Frame Position Table (based on the same frame positions for the source clip), and the new duplicate clip is set as the active (displayed) clip in the group. As in the case for the inverse clip and the forward clip, a video clip must be associated with the left facing clip. The video selected should be one which corresponds to the camera traveling on the same path as the forward clip but with the camera facing left.

STEP FIVE: Generate Right Facing Clip. Select an appropriate member of the group (which has the same direction as the right facing video) from the list of members in the right mouse button pop-up menu and then select "Create Duplicate Clip" from the right mouse button pop-up menu. If the Selected clip is not in a group already, this command allocates a groupID and adds a new record to the clip group table. The original forward clip is then assigned as a member of the group. A new record is added to the VideoClip and SpatialClip table for the duplicated clip, new FramePosition records are created (based on the same frame positions for the source clip), and the new duplicate clip is set as the active (displayed) clip in the group. As in the other cases the appropriate right facing video clip must be selected and associated with this clip in the now completed clip group.

STEP SIX: Are all paths done? The final step in the clip layout and video clip association process is to determine whether all members of the clip group are associated with various paths. If all paths in the video environment are defined and the video clips associated with each clip in each clip group, then this part of the process is completed. Otherwise repeat the steps, beginning at "Draw the first clip in a group" as shown on FIG.14, until done.

#### Associate and Edit Video

Where the paths are drawn on the map rather than generated from camera position data, the video clips must be associated with the map clips in the clip group represented by the drawn paths. The steps for a preferred embodiment of the Associate and Edit Video

process of the invention are illustrated in **FIG. 15** and described below. Where the map clips were drawn during the concept stage of the process, this method is:

STEP ONE: Select Clip. To select a clip on the map left-mouse-click on the line on the edit system map which represents the clip. The edit system responds to a successful clip selection by positioning a small red block which represents the camera position on the path. Since spatial video edit system only displays one clip in a group at a time, alternate members of the group may be selected by selecting the appropriate member, from those displayed in the right-mouse-click pop-up menu.

STEP TWO: Invoke Property Sheet. After selecting the clip, invoke the Clip property sheet by right-mouse clicking on the clip and selecting the Property command in the pop-up menu.

STEP THREE: Select Video Asset. The property sheet which is displayed by selecting the property command shows all of the fields in the Video clip table which correspond to the selected clip. See Attachment 1, page 19. In place of the video file field is a drop down list representing all the video assets currently identified in the asset database. Select the appropriate video asset which corresponds to this path. When the video asset is selected the other clip properties (MarkIn and MarkOut) are updated accordingly. Select the Apply or OK commands and the changes made to the property sheet are updated in the database.

STEP FOUR: Invoke Video Editor. To invoke the video editor (If one is not already displayed), right-mouse-click on the clip and select the Show Video command from the pop-up menu. See Attachment 1, page 20. This command displays the spatial video edit system's video editor.

STEP FIVE: Select In Frame. Use the video editor Play, Stop, Rewind, Step (Forward, or backward) command buttons to find the appropriate first frame in the clip which corresponds to this path. In this mode the video editor invokes the MPEG player to play the video, allowing inspection of the video and visual selection of the appropriate frame.

STEP SIX: MarkIn and Cut. Use the MarkIn button or the MarkIn dial to set the MarkIn to the currently displayed frame. Use the CutBefore MarkIn Button to set the MarkIn for the video clip relative to the video asset. This updates the database with the new MarkIn for the clip. As the edit system is based on database references, the "cuts" are not

made by removing any portion of the video assets, rather they are done by creating MarkIn or MarkOut references for a clip which refer to a particular video asset.

STEP SEVEN: Select Out Frame. Use the video editor's Play, Stop, Rewind, Step (Forward or Backward) command buttons to find the appropriate Last frame in the clip which corresponds to this path. In this mode the video editor invokes the MPEG player to play the video, allowing inspection of the video and visual selection of the appropriate frame.

STEP EIGHT: MarkOut and Cut. Using the MarkOut button or use the MarkOut dial to set the MarkOut to the currently displayed video frame. Use the CutAfter MarkOut Button to set the MarkOut for the video clip relative to the video asset. This updates the database with the new MarkOut for the clip. This completes the process for associating and editing Video.

### Identifying and Editing Intersections

Intersections are points on the map where 2 or more video clips or groups of clips intersect and it is desired to define video transitions which allow the player to traverse from one clip in the intersection to another during the spatial video production. Once such transitions are defined, the user of the final production has the option to execute the transition (usually a turn) at the intersection when playing the production. **FIG. 16** depicts an intersection of a north south clip group and an east west clip group. In a true spatial video map, the four paths of the group members of each group would be depicted by a single double thickness line. In **FIG. 16** the various clips in the clip group are expanded for purposes of illustration. A single clip group member could be accessed for editing or viewing as described above. The four component lines F (forward) L (left) R (right) and B (backward) are shown as four individual lines for each clip for ease of comprehension. **FIG. 13** shows an intersection **118** as it would appear on a map in the map editor.

The spatial video that the edit system preferably uses records scenes captured by a moving camera. For example, one clip might show the camera's view when crossing a room from north to south, while another clip may show a walk around the perimeter of the same room. The map window in spatial video edit system can represent each video clip by drawing the path that the camera took in capturing the clip. Of key importance to the production are the intersection points of these camera paths: places where one path crossed

or intersected with another. Each such point is an opportunity for the production to allow a smooth transition from one clip to another. If two camera paths intersect at right angles as in the north-south and east-west example illustrated in **FIG. 16**, the process herein described can allow a player moving north along the first path to turn right, and find himself going east along the second path. Of course, as seen in **FIG. 16**, there are a host of other turns which the editor could enable and allow a runtime user to choose.

There are two uses of clip exits to create smooth transitions in spatial video productions. One is the intersection clip exit, where the player makes a turn transition from one member of one clip group to a member of a different group. The other is the roundabout clip exit, where the player makes a turn transition from one member of a clip group to another member of the same clip group. These are discussed in more detail in the section on editing intersections below. Examples of turn transitions are the 2B-1B transition **110** and the 2F-1B transition **112** shown in **FIG. 16**. Examples of roundabouts are transitions 1F-1L **114** and 2B-2R **116**.

Typically a turn transition, such as the 2F-1B right turn **112** in the example discussed above, will be executed only when the player has expressed a desire to turn right. In order for this turn sequence to look realistic, it is important to choose carefully the exact frame from which we will leave the first clip, the exact frame in which we will begin the second clip, and the nature of the transition used for the turn. Poor choices here will result in a video sequence that jumps and tears during turns. These decisions can be guessed at by the software, but usually can be improved on by a human editor. For this reason, the system of the present invention offers the following services for intersections:

1. Automatic searching for clip intersections
2. Optional highlighting of intersections in a bright color
3. User review and specification of frames for departure and entry
4. User review and specification of transition type to use (e.g., "push left")
5. User preview of what the player will see when he makes the transition.

Identifying Clip Intersections. A number of approaches can be used for identifying the intersection points in a production. Since the camera paths are not constrained to be straight lines or simple curves, it is difficult to predict intersection locations without an exhaustive search. Quad tree technology would have allowed us to subdivide the area



covered by the map into successively smaller rectangles, and continue subdivision until each rectangle met one of the following criteria:

- the rectangle contained at most one camera path, or
- the rectangle was sufficiently small that all its points could be searched for intersections.

This quad tree technology produces a comprehensive database of all intersections in the production, but it requires considerable compute time. Moreover the resulting database would be invalidated every time a new clip was added, or an existing clip was moved. Such an invalidation might necessitate a repetition of the time-consuming process of searching for intersections: an outcome that might inconvenience the system user.

The preferred approach, therefore, is a dynamic one: identify only those intersections currently on the screen, and do it quickly enough that the system user does not have to wait. When the view is shifted to another region in the map, repeat the search for intersections in this new view (again so quickly that the user does not wait). When a clip is moved, or a new clip is added, repeat the search again. The search is done by successively drawing each clip into off-screen memory, using an additive coloring process that ensures any pixel that was colored twice ends up a different color from those colored once or not at all.

The steps to identify clip intersections in the current view using the preferred method are shown in **FIG. 17** and discussed below.

**STEP ONE: Allocate Image Buffer.** First, allocate a memory buffer into which paths can be drawn without them being visible on the screen. The edit system preferably uses twice the current resolution of the screen, so if the current map window is 400 x 300 pixels, the edit system will allocate a memory buffer big enough to hold an image 800 x 600 pixels.

The extra resolution reduces the chance of falsely identifying intersections at spots where two paths come close but don't touch.

**STEP TWO: Clear Image Buffer.** Second, clear the image buffer to all zeros.

**STEP THREE: Draw first path.** Third, draw the first path visible in the current view into the buffer using the first color, (say color "1"). All pixels in the buffer should now be zero if they are not on the first path, or "1" if they are.

For each remaining clip in the current view, repeat the following steps:

STEP FOUR: Draw new path. Draw the new path using a second color (say "2"). Use an additive graphics mode that adds (or otherwise combines) the new color onto the old. The result should be that each pixel in the buffer is zero if on neither path, one if on only an old path, two if on only the new path, but a different value ("3" in this example) if on both old and new paths).

STEP FIVE: Search Buffer. Search the image buffer for colors 2 and 3. Change all pixels that are 2 to 1. (They now represent an "old" path). Each pixel that is 3 indicates a new intersection. Add a description of this intersection location to a list of intersections found, then change the pixel back to 1.

STEP SIX: Repeat. Repeat steps four and five for the next path.

The list generated by the above process will likely contain duplicates. Depending on the current drawing scale and the widths of the camera paths, the overlap between two paths may comprise several contiguous pixels. Duplicates can be pruned from the intersection list either at the end, or each time a new entry is added.

Most of the compute time consumed by the above process is spent searching the image buffer for pixels of colors 2 and 3. It is therefore desirable to make this code efficient. Many processor architectures support fast searching for bytes or words of a particular value. In the Intel family, for example, the SCASB instruction results in a quick search and is a preferred embodiment. SCASB stands for "scan string bytes." It scans consecutive bytes in memory looking for a specified value.

Creating Transitions at Intersections. Where it is desired to allow the player to traverse from one clip to another at an intersection the edit system allows for the creation of a defined transition. For each "transition" there is a record in the ClipExit table to define the transition. The process for editing intersections in the preferred Microsoft Windows compatible format is as follows:

The user right mouse clicks on an intersection on the map, to invoke the video editor. A pop-up menu populated with all the reasonable choices of intersection transitions for all the video clips involved in the intersection is displayed. This list is computed as all the 2 pair combinations of video clips for each member of the video's in one group to members in the other group.

After the user selects which pair is to be used for the desired transition, the edit system looks in the ClipExits Table to determine if an intersection already exists between these two clips at a nearby frame. If an intersection already exists, then the user is prompted by a message box which pops up if the user wants to edit the existing intersection, add a new intersection, or cancel. If the user selects "Edit", then the intersection editor is initialized with the corresponding clipexit. If the user selects "AddNew" or if no intersection exists then the intersection editor (called the ClipExit Editor in the preferred embodiment) is initialized to edit a new intersection using the requested two video clips.

Initialization of the ClipExit Editor for New ClipExit. The ClipExit Editor is seeded with the two ClipIDs for the clips in the requested intersection, FromClipID, ToClipID. The ClipExit Editor then queries the map tool to determine the Frame numbers for the two clips at the intersection point. The frame numbers are used to initialize the clipexit editor which displays the two video clips, side-by-side with the "suggested" frames. There is a command button to swap the positioning of the two video frames so that the alignment of left or right turns produces a continuous image from one video to the next. The user then can fine tune or adjust the suggested frames by using positioning controls so as to produce the optimal image continuity. The user must also select the appropriate transition from the list of transitions. Typically this is the push left transition for left turns and the push right transition for right turns. The specific manner in which the runtime system creates this transition is disclosed in the co-pending Special Effects Applications.

However, teleportation type jumps or other requirements lead to opportunities for using other transitions as well. While some exits are seamless transitions, teleportation jumps occur when there is no smooth visual transition, such as between two spatially unrelated frames. The user must also select the appropriate signal to trigger the clipexit. This is specified in the ExitDirection Field. For example if the field is set to ExitLeft, then in the production a left turn signal (either using the mouse, keyboard, or onscreen control) will match with this clipexit and it will be activated when the number of the From Video frame is encountered.

The runtime system has a concept of QueuedExitID. This is a variable which indicates whether a clipexit should be taken when it is encountered. When the user moves the mouse to the left or right of the screen, the runtime system constantly updates a global

property named QueuedExitID. When the mouse is on the left of the screen Queued ExitID is set to LEFT; when it is on the right, it is set to RIGHT, and when it is in the middle it is set to DEFAULT. The clipexits all have a fixed unique ID and a common symbolic ID. The symbolic ID's refer to symbolic directions (left, right). These are usually mapped by flow definitions in the production from some user signal (e.g. move mouse left, click a button the left side of the screen) to set the QueuedExitID. Once the QueuedExitID is set by a flow, then the system will activate the exit if its exit direction matches. See text accompanying FIG. 22. Flow language and the runtime system are further discussed elsewhere in this specification.

Edit Roundabouts. As discussed briefly above, roundabouts are another type of clip exit. A roundabout is a location on the clip path where the player may execute a rotation. In the preferred embodiment, due to the use of anamorphic lens technology and panning discussed above, the player can pan anywhere within a 90 degree frame anywhere along the path. However, should the player wish to rotate or pan further than 90 degrees, visual information is needed from a second clip. A roundabout is a clipexit transition where the transition is from one clip of a group of clips to one of the two clips of that group of clips which contain visual information immediately adjacent to that contained at the left or right margins of the clip being navigated. For example, in a group of clips containing a forward, left, right and backward clip, roundabout transitions can occur from the forward clip to either the left or the right. Further, from the backward clip roundabout transitions can occur to either the left or the right clip. Of course, in a transition from the forward to the right clip, the forward clip will be panned fully to the right and the first transitional frames will contain information from the far left margin of the matched right clip frame. In contrast, where the transition is from the backward clip to the right clip, the first portions of the right clip to appear in a transition frame will be the rightmost portions of the right clip TO frame. Further, sequential roundabouts can create the effect of a 180, 270 or 360 degree turn. FIG. 13 shows a roundabout 120 as it would appear on a map in a map editor of the preferred embodiment. A roundabout symbol is also visible on page 18 of Attachment 1.

Notice that intersection clip exits by definition can only occur at the intersection of at least two paths. Roundabout clip exits, in contrast, can occur anywhere along the clip path, provided the developer defines them. The actual transitions in either case are performed

according to the combination panning and push transition methods disclosed in the co-pending Special Effects Applications previously mentioned.

Upcode or recode video. Another task which is preferably accomplished at edit time when MPEG or a similar reference frame system of digital video is being used is to upgrade all FROM frames in the video stream to I or P frames, and to create an I frame copy of all TO frames. If strict conformance to MPEG standards is desired, FROM frames must be I or P frames because they are used as reference frames during the transitions. The actual frame in the video stream must therefore be modified to be I or P if it was initially encoded as a B frame. If strict conformance to MPEG standards is not necessary, the method set forth in the co-pending Special Effects Applications can be used.

The copy of the TO frame that is present in the TO frame cache must be an I frame so that it can be decoded independently at the time the transition is to occur. The actual TO frame in the target video stream need not be changed, since the frame-accurate stream positioning technique described in the co-pending Special Effects Applications can be used.

Note that, in practice, the TO frame for one transition is likely to be the FROM frame for another transition. This is true, for example, of all frames involved in roundabout exits or intersection exits. In order to assure that the copies of the TO frames in the TO frame cache for each clip exit are I frames, some MPEG frames will need to be converted from B or P to I frames. This conversion process is called upcoding. Upcoding can be performed by decoding the selected frame into its luminance and chrominance values, then encoding those luminance and chrominance values as an I frame. In order to assure that the FROM frames in the source video stream for each clip exit are I or P frames, the video stream may need to be fully or partially re-encoded. Changing one frame from B to I or P will change the reference frames used to decode all frames between the changed frame and the following I frame, so the frame cannot just be modified and rewritten blindly. In order to accomplish the recoding, one of two methods are possible: A software re-encoder could analyze the stream, decode the appropriate frames, and re-encode them, or a list of FROM frames could be produced and the entire stream could be re-encoded by an encoder that accepts such a list. The preferred embodiment does neither of these, instead choosing the first of the three alternative implementations described below.

There are three alternatives to re-encoding the stream: First, the problem can be ignored. If this is done, any transitions that should have used a B frame as their FROM frame will actually use the I or P frame that would have followed the B frame in display order. The inventor has found that this produces acceptable results and eliminates the effort  
5 required for re-encoding or the other alternatives.

Second, the edit system could restrict the user from specifying a B frame as a FROM frame. This is not considered acceptable in most instances since it would be virtually impossible, given the desired B frame frequency, to assure that all four frames on all four clips involved in a roundabout or intersection would have I or P frames at the appropriate  
10 points. However, where video clips are generated by rendering computer models of a three-dimensional scene, the exact positions of each frame will be determined by the model and path used, so the technique could be applied.

Third, upcoded I frame copies of the FROM frames could be added to a FROM frame cache. When the transition takes place, the copy in the FROM frame cache could be used for  
15 the transition instead of the actual frame. This approach greatly increases the memory storage requirements at runtime, the overhead time spent on the clip, since the cached upcoded FROM frames would have to be read from the stream, and the size of the stream on the disk. Thus the second and third alternatives are not preferred.

Smooth transitions accomplished by the method of the co-pending Special Effects  
20 Applications required use of an MPEG player which can accept input from a buffer, rather than merely from a disk file. Further, they preferably are implemented through the use of a streamer, such as the MPEG streamer discussed above which is defined in Appendix C hereto. The MPEG Streamer function to play a transition is implemented in such a streamer by first requiring the streamer to execute an MPSChopStream call to terminate the MPEG  
25 stream at the current point. Next a valid GOP header is copied into the output buffer area. Third, the cached TO frame data is copied into the output buffer area. Finally the B frame data for the specified transition is copied into the output buffer area. This allows the streamer to create the "synthetic" MPEG stream of the transition as described in the co-pending Special Effects Applications.

## Hotspots

Hotspots are an important way to support user interaction in the product generated by the spatial video edit system. A hotspot is a region of the screen that the player can click on to make something happen. (Click on the vending machine and it produces a gum ball, click on the statue and the genie appears). There are two types of hotspots in spatial video productions. One hotspot is of fixed size and location on the display. These hotspots are called region hotspots to distinguish them from second type of hotspots, spatial video hotspots. Spatial video hotspots track features in the video and change in size and location from frame to frame. In contrast, region hotspots, as prior art hotspots, remain in a fixed position on the screen regardless of the navigational decisions made by the player. In spatial video productions region hotspots are often used in the user interface outside the video window to define exit buttons, inventory buttons and the like. They can also be used within the video window for navigation purposes. See, for example, **FIG. 27** and the text accompanying this figure. The screen navigation regions are essentially region hotspots which relate to panning and turn transitions. Unless specifically referred to as region hotspots, the references to hotspots in this specification are to spatial video hotspots.

In the present embodiment of the invention, hotspots are rectangular in shape, however hotspots can be any shape desired and remain within the invention. For example, triangular hotspots are an obvious variation. However, it is usually more computationally demanding to create hotspots of other shapes, therefore unless the production demands it, rectangular hotspots are preferably used. If one wants to respond to clicks on the statue, one defines a rectangular area that approximates the shape and size of the statue. Since the statue is not rectangular, this will result in small imperfections: either there will be small pieces of statue that are outside the hotspot, or there will be pieces of non-statue that are inside the hotspot (or both). In practice this seems to be unimportant: users tend to click the middle of the statue anyway. For shapes that are difficult to approximate with a rectangle (an equilateral triangle, for example), the spatial video edit system user can define a series of hotspot rectangles that collectively cover the desired shape.

The edit system user will want to indicate where in each scene the hotspots are, and what should happen as defined in the act/scene editor when the user clicks on each one. Since spatial video uses moving cameras, this task has the potential to be much more

difficult than it is for still-camera productions. A moving camera produces video clips in which everything changes position and size from one frame to the next. As the camera approaches the statue, the statue grows in every frame. As the camera moves left to pass the statue, the image of the statue moves more to the right in each frame. For a hotspot to track the statue, the hotspot must change in size and position in each video frame. The spatial video edit system user needs a simple way to make the hotspot follow the statue, without having to manually position it in every video frame.

To avoid the necessity of hand editing the hotspots in every video frame, both the spatial video edit system and the binder engine support the same interpolation/extrapolation algorithms that will take one or more user-specified hotspot frames, and "fill in" the missing ones to allow smooth size and position changes between the user-edited frames. The edit system user must manually position the hotspot in the first frame that it appears in. The edit system user must also identify the hotspot rectangle in the final frame (for the current clip) in which the hotspot appears. Editing of other frames for the hotspot is optional: the system will use linear interpolation for hotspot position and size between the user-edited frames. The edit system supports viewing the frames in which the hotspot appears, with the hotspot rectangle overlaid on the video in a bright color. When these frames are played successively (as motion video), it is easy to visually check the motion of the hotspot rectangle and see how well it tracks the clickable object (the statue in this example). If the rectangle wanders away from its target, the user can specify its position and size in any frame. Interpolation will then change to accommodate this new value. Typically the spatial video edit system user edits only a small fraction of the hotspot frames, finding that interpolation produces satisfactory hotspot sizes and locations in the remaining frames.

In a preferred embodiment, the binder performs the linear interpolation/extrapolation of the hotspots and creates a frame by frame list of hotspot locations and dimensions in the runtime data hunk. An alternate implementation lets the runtime system perform the interpolation rather than the binder. This requires increased CPU time during runtime, but reduces the memory needed for the list.

The temporal presence, activation, deactivation and other logic attributes of hotspots and receptacles are governed by flow logic, another aspect of editing which is discussed in more detail below.



To support hotspot editing, spatial video edit system provides these services:

1. Random access to any frame in any clip
2. Definition of a hotspot rectangle in any frame by simple point-and-click operation
- 5 3. Visual feedback showing the hotspot location as a rectangle overlaid on the video
4. visual feedback as to whether the current frame has an interpolated rectangle, or a user-specified rectangle
- 10 5. tuning of the size and location of hotspot rectangle in any frame by dragging the center , edges, or corners of the rectangle
6. Deletion of user-specified rectangle values (causing the current frame to revert to interpolation)
7. The ability to play the hotspot frames as motion video, with the hotspot rectangle overlaid in each frame
- 15 8. The ability to color segments of the clip's path in the map to show which path frames include rectangles for the current hotspot
9. Storage of the resulting hotspot data in the edit system database for subsequent use by the binder.

20 In the preferred embodiment of the edit system, the system allows the play of the hotspot frames to be played backwards or forwards using the methods disclosed in the co-pending Special Effects Applications. Storage of rectangle positions and sizes are done using the coordinate space of the source video stream. For example, if the frames in the source are 352 x 240 pixels, then all rectangle measurements will run 0-351 in the X dimension, and 0-239 in the Y dimension. Relating the measurements back to the source  
25 (rather than to the current display, for example) produces numbers that are still useful when the video is later displayed at a different magnification.

It is important to choose for this operation a video player that supports the ability to draw rectangles on top of each video frame, both while paused, and while playing full motion video. In the preferred embodiment the edit system uses the SoftMotion player  
30 available from SAS Institute Incorporated. When the user edits a video clip, the edit system fetches from the database all information regarding hotspot rectangles currently defined in

the clip. This information is kept in memory, where it is available for quick access any time the video editor is showing a frame that may contain hotspots.

Whenever a frame is displayed, the in-memory copy of the hotspot data is searched for rectangles that need to be drawn. The position and size of any hotspot rectangle in the frame will be determined in one of two ways. If the rectangle was edited by the user for that frame, then exact values, as specified by the user, will have been stored in the database, and are available in the above mentioned in-memory copy of that data. Other frames will require interpolation. Interpolation will be explained below. Once the position and size of the rectangle are known, the rectangle may be drawn on top of the video, optionally using one color for user-specified points, and a different color for interpolated points.

Similarly, wherever a camera path is drawn on the map, the color of that path may optionally highlight those frames that contain rectangles for the current hotspot. Since the database records the first frame and last frame for each hotspot in each clip and each point in the camera path is drawn in one color if its frame number lies between the first and last hotspot frames and a different color otherwise, this is a straightforward operation.

The video editor supports a variety of editing modes. The current mode may be changed by the user through the edit system's push buttons and menu system as described above and further illustrated in Appendix A. In the default edit mode, it is assumed that hotspots are not being edited. They are all drawn in the same color, regardless of whether their rectangles are interpolated or user-specified. In the default mode no new hotspot can be added. In this mode the user may select a hotspot by clicking on its rectangle. The name of the currently selected hotspot is shown in the window caption, and the segment of the camera path that contains that hotspot is colored in the map (if that option is enabled).

A special edit mode is used for changing existing hotspots. For this discussion it will be called the "change hotspot" mode. In this mode, the system tracks one hotspot as the one currently being edited. All other hotspots are drawn in the same color as they were in the default edit mode. The rectangle for the selected hotspot is drawn in one color for a user-specified rectangle, and a different color for an interpolated rectangle. The user can drag any edge or corner of the rectangle to change its size, or drag any interior point to move the entire rectangle without changing its size. Any rectangle that is so adjusted becomes a user-specified rectangle (not an interpolated rectangle). The database is updated to reflect the new

size and position, and the color of the rectangle is changed (if necessary) to the color used for user-specified rectangles.

In the "change hotspot" mode, the user can use the standard video player controls to move to any other frame of video, thus allowing him to see the hotspot rectangle in any frame. If the user positions to a frame outside the range of frames currently containing the hotspot, the hotspot's range will be extended to include that frame. The user can also play the frames as full motion video, with the hotspot rectangle overlaid in each frame. This allows the user to determine whether the rectangle moves appropriately to track the scene element that the user will click on.

A third edit mode is selected to add a new hotspot to the video. For this discussion it will be called the "add hotspot" mode. It is entered when the user requests it with no hotspot currently selected. This mode is used to establish the first data point for a hotspot. In this mode the user drags out a rectangle on the video to define the hotspot location in the current frame. When the rectangle is completed the user will automatically drop into the previous ("change hotspot") mode where he can adjust the new rectangle and extend the range of the hotspot to other frames.

Interpolation is an important aspect of this process. Although the database contains descriptions of only the user-specified frames, the editor preferably keeps a table in memory with hotspot size and position information for all video frames. Because this table describes every frame in a hotspot's range, no new calculations need be done when drawing hotspot rectangles on a new video frame: the answers are already in the table. This quick response is important when playing full motion video. Each frame description in the table contains a boolean field to distinguish user-specified values from interpolated values. This is important because a single user edit operation (like moving a hotspot rectangle in one frame) could invalidate the table entries for a whole series of frames. After every such action, the edit system re-interpolates the hotspot data and updates its table in memory.

The first step in preferred re-interpolation process is to discard all values that were not user specified (i.e. that were the results of previous interpolations). The edit system then processes the user-specified frames in pairs, considering the first and second, then the second and third, then the third and fourth, and so on. If the pair represent adjacent frames of video (frame numbers differ by one), then there are no intermediate values required between them.

In that case the system moves on to the next pair of user-specified frames. Otherwise the system generates values for all the frames between the current pair as follows. Simple linear interpolation is used for the X- and Y-coordinates of the rectangle center. Simple linear interpolation is also used for the height and width. For example, suppose the X-coordinate of the center moves from 11 to 21 when going from frame 100 to 105. The edit system allocates the total movement (10) evenly over the intermediate frames, producing values of 13, 15, 17, and 19. So the entire sequence for frames 100 to 105 would be 11, 13, 15, 17, 19, and 21. While more advanced forms of interpolation (like splines) may produce smoother motion, linear interpolation requires less computation, and produces good results. Linear interpolation is an easy concept to explain to a new spatial video edit system user. It does not have the (possibly surprising) side effects that spline-based interpolation would have such as overshooting values beyond control points.

#### Object Classes and Object Instances

The word "object" has been used herein according to its ordinary meaning to describe objects in the video environment as well as objects overlaid on the video environment, such as M-views. However, there is also a technical meaning of the terms "object class" and "object instance" as they are used in the preferred spatial video edit time and runtime systems to describe properties of overlays.

It is important to appreciate the difference between the concepts of object class, object instance and M-views. The edit time system supports the concept of an object class. An object class is a definition of a collection of variables that describe the attributes and behaviors of a user-defined object such as a "ball" or "inflatable object." Characteristics common to all members of the class are described in the object class definition in the edit time database. Such characteristics are described by the flows that control the object's behavior and state variables that describe the object's current state.

An object instance is created from the object class definition and, in addition to containing a reference to the object class, is represented by its own collection of variables which are distinct from any other object instance created from the same class. For example, there can be multiple ball instances created from the ball object class; each ball instance can have its own appearance, location and other attributes. While the flows associated with an

object are generally contained in the object class definition, object instance attributes which may be unique to a particular object instance can be used by the object class flows to determine how the particular object instance reacts. All required object instances for a production are specified during editing time and space for their variables, including pointers to the object class, is allocated in the data hunk at bind time. The object class definition and flows associated therewith are in the data hunk and are available to the object instances.

Object instances can be placed in a receptacle. A state variable in the object instance tells which receptacle. Object instances have an appearance. Another state variable in the object instance tells which M-view represents the current appearance. Object instances may have other state variables defined by the editor such as "current inflation status." Flows change state variables by assigning new values to them. For example, when two object instances interact, such as the ice pick in the users "hand" and the basketball in the table receptacle, both the ice pick and the basketball have flows which run and cause one or both object instances to change state. The ice pick may have a "sharpness" state that decreases and the basketball's inflation state and appearance may change.

Use of the object class and instance concept for creating objects allows considerable conservation of programming resources by allowing the developer to refer to the object class rather than create new properties and behaviors for common characteristics of class members used in the production. This approach to the design of objects to be placed in receptacles in the environment also makes it computationally much easier to move objects from one receptacle to another without fragmenting object or receptacle information. Flows can also be associated with an object to govern various types of interactions with other objects or with receptacles. Flows can be triggered by synchronous or asynchronous events.

The edit time and the runtime systems support a collection of variables known as the "object instance." An object instance is the individual object class member that may appear in and/or interact with the production. At edit time an object instance may refer to the object class variable for various characteristics. At bind time, for runtime computation efficiency, the object class variable information is copied directly into the runtime object instance variable. Thus, in the preferred embodiment of the system object class variables do not exist in the final runtime product, only object instance variables are present.

An object instance is a created object with an assigned initial location, flows and other variables. Object instances consist of a collection of data items ("properties") which specify such things as the M-view associated with the object when it is displayed in the spatial video, cursors used to represent the object after the user has picked it up, flows to run when the object interacts with other objects, and so forth. The system allows the user to add data items associated with objects.

An object instance variable either at edit time or at runtime has several properties associated with it, including the M-view that is used to represent its current state. When the state of the object instance changes, the same action which causes a change in state could trigger a flow which changes the M-view associated with the object instance. Thus a single M-view could be associated with several different object instances. For example, there could be several inflated basketballs at different locations in the environment. Each individual basketball would be an object instance. However, a single inflated basketball M-view could be associated with all of these object instances. This also allows for computational efficiency.

A different M-view may be associated with another object instance of the same general object. For example, the basketball object class could contain a blue basketball subclass and one or more blue basketball object instances as well as an orange basketball subclass and one or more orange basketball object instances. Blue basketball M-views could be associated with the various states of the blue basketball object instances. Orange basketball M-views could be associated with the various states of the orange basketball object instances.

As an example of how object instances are used, suppose a production has a basketball object instance in an inflated current state with a current position on a table in a room of the production environment. The M-view associated with the object instance depicts an inflated basketball. Further, suppose the user has previously picked up an ice-pick object instance. The user's cursor would be a representation of the ice pick. If the ice pick is clicked on to the inflated basketball, the basketball should deflate. The user would hear a hissing sound, caused by a flow associated with the interaction of the ice pick and the basketball which triggers the playing of audio clip of a hissing sound stored as a .WAV file or other sound file, and the basketball object instances' M-view would be replaced by a burst

basketball M-view. The change in the basketball's state would probably need to be noted elsewhere, probably in some sort of state variable in the basketball object instance.

As a result of the interaction of the basketball object instance with the ice pick object instance, the basketball object instance would be changed in that the state would now be set at burst, the M-view and cursor associated with that object instance would be correspondingly altered.

### Placing M-views / Receptacles in Video

An important feature of the spatial video production system is the ability to augment a video scene with bitmaps of objects. Footage of a bare tabletop can, at run time, can show a table with a vase on it, or a frog, or any other bitmap image. Information about what object to draw, where to draw it, and how to draw it must be entered during editing so that it can be fed to the run time engine when needed. The entire operation would be simple were it not for the fact that spatial video uses moving cameras. As the camera approaches the table, the viewer expects the object on the table to grow in size on each frame, just as the image of the table grows in each frame. And as the camera begins to go past the table to one side, the angle that the object is viewed from must continually change.

The moving camera makes the job more difficult in two ways. First, we must be able to draw each object in varying sizes and from varying points of view. This aspect is covered in another section of this document. Second, we must supply unique location, scale, and angle data for each object in each frame. The spatial video edit system supports the creation, review, and modification of this data.

The term "receptacle" is used to refer to a location in a video clip where an M-view may be drawn. To avoid the necessity of hand editing the receptacles in every video frame, both spatial video edit system and the binder support the same interpolation algorithms that will take one or more user-specified receptacle frames, and "fill in" the missing ones to allow smooth scale factor, position, and angle changes between the user-edited frames. The spatial video edit system user must manually position the receptacle in the first frame that it appears in.

The spatial video edit system user must also identify the receptacle location in the final frame (for the current clip) in which the receptacle appears. Editing of other frames for

the receptacle is optional: the system will use smooth interpolation for receptacle position, scale factor, and angle between the user-edited frames. The edit system supports viewing the frames in which the receptacle appears, with the receptacles represented in the video both by a rectangle, and by an arbitrary M-view object. When these frames are played successively  
5 (as motion video), it is easy to visually check the motion of the M-view object and see how well it tracks the background. If the M-view wanders with respect to the video background, the user can specify its position, scaling factor, and angle in any frame. Interpolation will then change to accommodate these new values. Typically the edit system user edits only a small fraction of the receptacle frames, finding that interpolation produces satisfactory  
10 receptacle appearance in the remaining frames.

During the edit process, a receptacle will be represented on the screen in two ways: both as a rectangle, and as an M-view. The M-view lets the system user see the end result (what the ultimate product user will see). The rectangle gives a convenient graphical control for changing location and scaling factor. (The size of the rectangle is proportional to the  
15 current scaling factor).

To support receptacle editing, the spatial video edit system provides these services:

1. Random access to any frame in any clip
2. Definition of a receptacle in any frame by simple point-and-click operation
3. Selection of a "typical" M-view to show in the receptacle
- 20 4. Visual feedback showing the receptacle location as a rectangle and M-view overlaid on the video
5. Visual feedback as to whether the current frame has interpolated receptacle values, or user-specified ones
6. Tuning of the receptacle position and scale factor in any frame by dragging  
25 the center, edges, or corners of the rectangle
7. Tuning of the angle of view by simple mouse actions
8. Deletion of user-specified receptacle values (causing the current frame to revert to interpolation)
9. The ability to play the receptacle frames as motion video, with the receptacle  
30 rectangle and M-view overlaid in each frame



10. The ability to color segments of the clip's path in the map to show which frames include values for the current receptacle
11. Storage of the resulting receptacle data in the spatial video edit system database for subsequent use by the binder.

5 Storage of receptacle positions is done using the coordinate space of the source video stream. For example, if the frames in the source are 352 x 240 pixels, then all position measurements will run 0-351 in the X dimension, and 0-239 in the Y dimension. Relating the measurements back to the source (rather than to the current display, for example) produces numbers that are still useful when the video is later displayed at a different  
10 magnification. For more precise positioning, fractional values can be used. It is entirely reasonable to store an interpolated X-coordinate, for example, as 91.5. Depending on the magnification used during playback, and the anti-aliasing algorithm employed, a value of 91.5 could produce a different (and more realistic) result than either 91 or 92 would have done. Scale is stored as an arbitrary multiplication factor, and angle is stored in degrees.

15 It is important to choose for this operation a video player that supports the ability to draw rectangles and M-views on top of each video frame, both while paused, and while playing full motion video. The preferred player is the SoftMotion player previously described. When the user edits a video clip, spatial video edit system fetches from the database all information regarding receptacles currently defined in the clip. This information  
20 is kept in memory, where it is available for quick access any time the video editor is showing a frame that may contain receptacles.

Whenever a frame is displayed, the in-memory copy of the receptacles data is searched for rectangles and M-views that need to be drawn. The position, scaling factor, and angle of any receptacle in the frame will be determined in one of two ways. If the receptacle  
25 was edited by the user for that frame, then exact values, as specified by the user, will have been stored in the database, and are available in above-mentioned in-memory copy of that data. Other frames will require interpolation. Interpolation will be explained below. Once the position, scaling factor, and angle of the receptacle are known, the rectangle may be drawn on top of the video, and an M-view may be rendered on the frame using these values.

30 Similarly, wherever a camera path is drawn on the map, the color of that path may optionally highlight portions of the path corresponding to frames that contain the current

receptacle. Since the database records the first frame and last frame for each receptacle in each clip, this is a straightforward operation.

The video editor of the system supports a variety of editing modes. The current mode may be changed by the user through the menu system. In the default edit mode, it is assumed that receptacles are not being edited. Their rectangles are all drawn in the same color, regardless of whether their values are interpolated or user-specified. In the default mode no new receptacle can be added. In this mode the user may select a receptacle by clicking on its rectangle. The name of the currently selected receptacle is shown in the window caption, and the segment of the camera path that contains that receptacle is colored in the map (if that option is enabled).

A special edit mode is used for changing existing receptacle positions and scaling factors. In this mode, the system tracks one receptacle as the one currently being edited. All other receptacles are drawn as they were in the default edit mode. The rectangle for the selected receptacle is drawn in one color for user-specified values, and a different color for interpolated values. The user can drag any edge or corner of the rectangle to change the receptacle's scaling factor, or drag any interior point to move the entire rectangle without changing its scaling factor. Any rectangle that is so adjusted becomes a user-specified rectangle (not an interpolated rectangle). The database is updated to reflect the new size and position, and the color of the rectangle is changed (if necessary) to the color used for user-specified rectangles.

In the "change receptacle" edit mode, the user can use the standard video player controls to move to any other frame of video, thus allowing him to see the M-view and rectangle in any frame. If the user positions to a frame outside the range of frames currently containing the receptacle, the receptacle's range will be extended to include that frame. The user can also play the frames as full motion video, with the M-view and rectangle overlaid in each frame. this allows the user to determine whether the M-view moves appropriately to track the video background.

A third edit mode is selected to change the angle of view. In this mode, the mouse is dragged to the left or right to select different angles of view for the M-view. The screen is updated accordingly so that the user can see the results.

A fourth edit mode is selected to add a new receptacle to the video. In this mode, the first click on the video window will be taken as the initial location for the receptacle in that clip. An arbitrary scaling factor will be chosen, and the user will automatically drop into the previous ("change") edit mode where he can adjust the new rectangle and extend the range of the receptacle to other frames.

Interpolation is an important aspect of this process. Although the database contains descriptions only of the user-specified frames, the spatial video edit system keeps a table in memory with location, scale factor, and angle of view for all video frames. Because this table describes every frame in a receptacle's range, no new calculations need be done when drawing a new video frame: the answers are already in the table. This quick response is important when playing full motion video. Each frame description in the table contains a boolean field to distinguish user-specified values from interpolated values. This is important because a single user edit operation (like moving a receptacle rectangle in one frame) could invalidate the table entries for a whole series of frames. After every such action, spatial video edit system re-interpolates the receptacle data.

The first step in re-interpolation is to discard all values that were not user specified (i.e. that were the results of previous interpolations). The spatial video edit system then processes the user-specified frames in pairs, considering the first and second, then the second and third, then the third and fourth, and so on. If the pair represent adjacent frames of video (frame numbers differ by one), then there are no intermediate values required between them.

In that case spatial video edit system moves on to the next pair of user-specified frames. Otherwise spatial video edit system generates values for all the frames between the current pair as follows. Simple linear interpolation is used for the X- and Y-coordinates of the receptacle. Simple linear interpolation is also used for the scale factor and angle of view.

For example, suppose the X-coordinate of the center moves from 11 to 21 when going from frame 100 to 105. spatial video edit system allocates the total movement (10) evenly over the intermediate frames, producing values of 13, 15, 17, and 19. So the entire sequence for frames 100 to 105 would be 11, 13, 15, 17, 19, and 21. While more advanced forms of interpolation (like splines) may produce smoother motion, linear interpolation requires less computation, and produces good results. Linear interpolation is an easy concept to explain

to a new spatial video edit system user. It does not have the (possibly surprising) side effects that spline-based interpolation would have.

### Flows

5 Flows are used in spatial video productions to allow the developer to specify execution logic for the product. A flow may say what happens when the product user clicks on hotspot X, or when the video player gets to the end of clip Y. The logic in a flow is specified in a simple programming language that may be entered and changed in the logic editor portion of the Spatial Video Edit System. Once entered the flow logic will be scanned  
10 by the edit system for obvious errors or omissions. Further error checking will be done by the system binder (discussed below) when the executable final production is generated. At bind time the flow language is translated into its final form to allow efficient playback.

All flows start with a FLOW statement and end with an ENDFLOW statement. Between these are a list of executable statements like assignments, loops and conditionals.  
15 The syntax for these statements is contained in the Flow Language Reference contained in Appendix A.

The preferred spatial video edit system provides the user with the ability to define global variables for use within flows. These variables are called "properties" and are defined and initialized in the edit-time database. The preferred system allows four types of  
20 properties: numeric, boolean, string and memo. Numeric properties are four-byte signed integers. Booleans are one byte yes/no values, with 0 meaning no. String properties are preferably fixed length 255 byte strings of characters. Strings shorter than 255 bytes are padded to 255 bytes with blanks. Memo properties are variable-length strings preceded by a two-byte length code. The user of the system can write flows which refer to these properties.

### Logic Editor

The logic edit steps of the edit process are those which associate various logic flows with the components of the production. As the logic programming is text based, it can be edited with any simple text editor. Such an editor is built into the preferred Spatial Video  
30 Edit System.

Embedded external programs. There are certain types of programming that are not efficiently written in the flow language. An alternative is to construct a separate special purpose program which “owns” a region of the display screen for some period of time. The region “owned” by the special purpose program may overlap with other visuals and a priority means may determine which visual is visible at what time. Such programs can be debugged and tested separately from the overall production. One example of such a program might be a visual which portrays an automobile instrument cluster in which the speedometer and tachometer reflect changes in rpm and speed. This program could occupy a lower part of the display while a video window was showing motion video through the “windshield” above the instrument cluster. Such special purpose programs should be constructed with a well defined interface which allows them to interact easily with the flow system. In the preferred Microsoft Windows environment, such programs can be constructed as OCXs. These program types are documented in the *Microsoft Developer Network Development Platform SDKs and Operating Systems* documentation previously cited. Similarly, OCX’s can be used for programs without visual presentation, such as subroutine libraries.

### Bind Step

At the end of the edit process the bind step occurs. This step invokes the binder to create the final multimedia spatial video production. The binder is the component of the system which reads the database produced by the spatial video editor and writes disk files in a format that can be quickly and easily loaded and interpreted by the runtime system. It is similar in function to a compiler and linker in a traditional edit-compile-link-run software development environment.

In a preferred embodiment, the binder reads the database and collects the data therein into three main data files, as well as some smaller auxiliary files. The three main files are known as the “data hunk,” the “logic hunk,” and the “clip hunk.” The **FIG. 28** illustrates the overall binder process, containing the steps of data hunk creation, logic hunk creation, initialization file creation, clip hunk creation, and perform relocations.

The clip hunk preferably is a single large MPEG file consisting of all the spatial video in all the clips in the production. Each clip is preferably preceded by a number of I frames which represent other frames that can be reached by means of clip exits present on

the clip. These frames are present to allow the runtime system to cache them before beginning to play the clip. Each clip is preferably followed by a single copy of the last frame of the clip, but the copy is always an I frame (regardless of the actual type of the last frame).

This ensures that the future buffer never contains any inappropriate frames and any transitions from the end of the clip elsewhere look correct.

The data hunk preferably contains all the information the runtime system needs to know to set up the hotspots, receptacles, visuals, global properties, music, and asynchronous (keyboard and mouse, joystick, etc.) input for the production. The data hunk is derived from the edit-time database and is laid out by the binder in a fashion which makes it easy for the runtime system to allocate and initialize the memory needed to store and represent this data while the production is running. For the properties (discussed above), the binder must do some special work to assign each property an offset in the data area of the "runtime machine" which is described below. The property offsets are used to refer to the properties in the runtime flow language interpreter. **FIG. 29** illustrates the steps of the data hunk creation process.

The logic hunk (runtime logic file) is a collection of instructions in a low-level language that is interpretable by the runtime system, and a list of events that should occur when a given frame of video is displayed (known as "frame events"). The instructions implement general-purpose programming commands for moving and comparing data, performing arithmetic, playing audio and video, and so forth. The frame events tell the runtime system when to activate and deactivate hotspots and receptacles, when to play certain music files, when to activate flows and clip exits, and so forth. The binder generates the logic hunk by reading the flows written by the spatial video edit system user and generating the low-level runtime language instructions that implement the user's commands.

It generates the frame events by examining the entries in the database pertaining to clip exits, frame positioning, hotspots, receptacles, and other information and writing corresponding frame events to the logic hunk each clip in the production. The logic hunk creation process is illustrated in **FIG. 30**. For an example of low-level flow language instructions see **Table 4b**. For an example of a series of frame events, see **Table 3**.

In addition to the three main generated data files, the binder creates an initialization file which tells the runtime system where to find the other files involved in the production.

For example, if cursors, music, digitized audio files or dramatic video files are used, the initialization file gives the path name to allow the runtime system to find these files. The binder also copies over to the production certain files containing other assets used in the production, such as dramatic video files, audio files and MIDI files.

5           When instructed to do so, the binder can also copy these various assets into the same directory tree in the file system that the binder-created files occupy. This has the effect of removing all dependencies on files outside the binder-created directory tree, which makes it very convenient to create a CD-ROM that will run the production without encountering problems due to different disk names or drive letters. When the binder runs in this mode,  
10           only path names relative to the program executable are used.

          The binder makes use of a technique used commonly in the art of linker design called "relocation." Relocation allows the various parts of the data and logic hunks to refer to items in the other hunks by means of a numerical offset into the hunk, even though the offset is not known at the time the reference is written. This is accomplished by storing a list of positions  
15           in each hunk that are in fact references to another hunk, then rewriting the data at those positions after the other hunk has been created. For example, if a hotspot in the data hunk refers to a flow in the logic hunk, it cannot know the offset in the logic hunk at which the flow begins; therefore, a zero is written to the data hunk as a placeholder and the offset is noted. Later, after the logic hunk is written, the data hunk is reopened, a seek is used to  
20           position the data hunk file to the correct location, and the correct logic hunk offset is written to the file, overwriting the zero that was previously written as a placeholder.

          One portion of the bind process is that used for creating frame events on a specific clip:

Step One: If the current clip has a MIDI file associated with it, generate a  
25           PLAY\_MIDI frame event for the first frame of the clip.

Step Two: For each clip exit which specifies the clip as its FROM clip, do the following:

- a.       If the clip exit specifies a flow to run, generate a CLIP\_EXIT event for the frame associated with the clip exit that calls the given flow;
- 30       b.       If the clip exit specifies a valid TO clip, do the following:

- (1) Generate a CACHE\_FRAME event associated with the first frame of the clip that caches the TO clip target frame;
- (2) Generate a RUNTIME\_TRANSITION event associated with the clip exit frame that specifies the runtime transition to use when taking the clip exit;
- (3) Generate a CLIP\_EXIT event associated with the clip exit frame that specifies the TO clip and frame number to exit to;

Step Three: For each receptacle which is present on this clip, generate a RECEPTACLE\_ACTIVATE and a RECEPTACLE\_DEACTIVATE event for each time the receptacle becomes active or goes inactive. The events should be associated with the frame on which the receptacle becomes visible or becomes invisible, respectively. If the receptacle is visible when the clip starts, the RECEPTACLE\_ACTIVATE should be associated with the first frame of the clip, similarly, if it is still visible when the clip is exited, the RECEPTACLE\_DEACTIVATE should be associated with the last frame of the clip.

Step Four: For each hotspot which is present on this clip, generate HOTSPOT\_ACTIVATE and HOTSPOT\_DEACTIVATE events similar to the RECEPTACLE\_ACTIVATE and RECEPTACLE\_DEACTIVATE events of step three;

Step Five: Generate a CLIP\_END event associated with the last frame;

Step Six: Sort all frame events based on the frame with which they are associated. If two events are on the same frame, make sure that RUNTIME\_TRANSITION events immediately precede their associated CLIP\_EXIT event and that the CLIP\_END event comes last.

Note that it is possible for the binder to change some basic parameters of the production. For example, the binder could, as part of the bind process, translate all MPEG video to AVI, Indeo, or some other format. If a new version of the runtime system were to be implemented using the new format, the user could generate working productions for both formats by simply binding the same database in two different ways. Furthermore, the binder could bind the production differently for different target platforms. For example alignment and byte ordering are different on computers based on the Motorola series of processors than they are in the Intel processor series, but a binder running on an Intel platform could reorder bytes as appropriate and create a production that would be easily readable by a runtime



system running on a Motorola platform. Neither of these translations, nor any others that the binder might implement, would require changes in the Spatial Video Edit System database.

### Runtime Step

5           Once the edit step is complete the relevant portions of the database and assets are bound into the final production. The runtime system engine must be able to coordinate playing video with overlaying of M-views on the video, playing sounds, and interacting with the production with mouse or keyboard.

10           The runtime system is preferably an interpreter-based system. While a compiled system would also work, an advantage of using an interpreter system is its ability to represent known classes of complicated ideas very compactly. At runtime, this can provide significant efficiencies. Whichever type of system is used, it must be able to run flows and play video sequences with overlaid hotspots and receptacles. The runtime system also coordinates external events from keyboard, mouse, or other input devices.

15           The runtime system's primary interpreter is the flow interpreter. There is also a video interpreter as will be described below. The flow interpreter and video interpreter work in parallel with the MPEG stream in a logic stream which allows the runtime system to determine, on a frame by frame basis, whether anything should be overlaid on the frame about to be displayed. Each frame of video is decoded from the MPEG stream into a buffer.

20           After decoding, the buffer holds a bitmap of the image that is about to be displayed. Before the image is displayed any pending frame events or other events are handled. The classes of events are described in this runtime section and the appendices incorporated herein. They are illustrated in the figures accompanying the text. After the events are handled, any active receptacles are handled by determining for each active receptacle whether there is an M-view in the receptacle and, if so, determine the receptacle's position and the M-view's size and angle from the active receptacle list. The M-view is then rendered into the buffer holding the bitmap of the image to be displayed. The transparent parts of the M-view are not drawn, only the opaque parts. While the M-view could be rendered directly onto the display, the decompression, scaling, locating and rendering of the M-view bitmap is not instantaneous.

25           Consequently, drawing them directly on the display could result in screen flicker. Therefore,

in the preferred embodiment, the M-views are rendered on the bitmap of the decoded video frame while it remains in the buffer and before it is displayed.

The overall runtime system process is shown in **FIG. 18**. When the run time system begins it loads the data hunk using the steps described below, and then begins interpreting the initial flow. The flows all come from a file called the logic hunk. The logic hunk contains a number of programs that control the course of the production. These programs are sometimes referred to as flows and were the programs created by the logic editor of the edit system. The data hunk contains information about locations of hot spots, and receptacles, within frames. The data in the data hunk and the logic flows in the logic hunk were incorporated into the runtime data hunk and logic hunk during binding as described above. An MPEG file, called the clip hunk, contains the spatial video footage. An M-view file contains the bitmaps (faces) of the objects which are inserted into the video. Other files include midi music, dramatic video sequences, and digitized sound.

The initialization process preferably loads the data hunk into memory using the following steps. In the first step MIDI file information is loaded and indexes assigned based on the MIDI file information in the data hunk. Next, Visual information is loaded, namely, Windows are loaded and indexes assigned, Viewports are loaded and indexes assigned, Pictures, or visuals which frame bitmaps, are loaded and indexes assigned and, finally, custom OCX visuals are loaded and assigned indexes. All index assignments are based on the order of the indexed data in the data hunk.

After visual information is loaded, receptacle information is loaded. First receptacle lists and associated flows, if any, are loaded and indexes assigned. Then the list of receptacle positions, viewing angles, and scaled sizes on each active frame and any associated flows is loaded and each list is associated with a receptacle.

After receptacle information is loaded, hotspot information is loaded. First the list of hotspots and their associated flows are loaded and assigned indexes based on their order in the data hunk. Then the list of hotspot positions and sizes on each active frame is loaded and each list is associated with a hotspot. After hotspot information is loaded, information on how to handle asynchronous user input is loaded. this information consists of a list of input events and a flow associated with each event. After input events are loaded, global properties, constants, and objects are loaded.

The logic hunk is much simpler to load, since it merely consists of a large block of logic instructions. It can be loaded in one pass by allocating a single block of memory and filling it with the contents of the logic hunk file. As the logic offsets are included in the various pointers in the data hunk, no index need be constructed for the logic hunk.

5        Execution proceeds by allowing the first logic hunk flow to run. See **FIG. 18**. In the preferred embodiment this program builds a user interface and starts a video running. The user interface is built by loading a series of visuals and making them appear on the screen. The visual typically starts the first spatial video sequence. The interpreter manages the flows and the spatial video. Among the first events in a spatial video clip are those which load  
10        cached frames into memory. These are the TO frames for the various transitions permitted from different frames within the clip. These cached frames are preferably placed in the video stream at bind time to be loaded when the video clip begins to be played. When a clip is played from a starting frame other than the beginning frame, these cached frames are loaded into a cache frame buffer by the MPEG streamer in response to the DOEVENTS catch-up  
15        mode routine accessed through the video operation code (op code) interpreter as illustrated in **FIGS. 20** and **22**, and as discussed in the text accompanying these figures.

It is easiest to understand the separate but related functions of flow management and video management by considering the interpreter as two interpreters, a flow interpreter, i.e. a recursive op code interpreter, and a video op code interpreter. The video op code interpreter  
20        manages the playing of the frames but passes to the flow interpreter when events call for any flows. See **FIG. 20** and **FIG. 19**. These functions will be discussed in greater detail below.

Associated with the MPEG video is a section of the logic hunk that is keyed to particular frame numbers in the video. This logic hunk section is pre-sorted by the binder in ascending frame number order and contains events for any receptacles and hotspots which  
25        were associated with those video frames at edit time. The runtime system refers to the location information loaded from the data hunk and overlays the hotspots and receptacles on the proper locations on the proper video frames.

Each hotspot and receptacle can have a flow assigned to it for each mouse event (or other user input device action). For example the associated flow logic could select different  
30        cursors to display when the mouse enters and leaves the hotspot. Another example would be to have the left mouse button execute a help program if clicked when the cursor is over the

hotspot. The types of actions associated with mouse events in multimedia runtime systems are well known to those skilled in the art. However, the uses of a runtime system to track hotspots and receptacles in spatial video and to interpolate size and angle of M-views associated with the spatial video are novel aspects of the system.

5 When a receptacle activate event is triggered, an M-view that is associated with the object in the receptacle will be displayed at a location and size as described by the data file for each subsequent frame of video with which that receptacle is associated. The data in this data file is based on the receptacle definition data created at edit time, however the binder not only places the position and size data from the edit time database into the runtime data file, it  
10 preferably also interpolates frame-by-frame at bind time and places the interpolated data in the data file to minimize the computational work to be done at runtime.

Hotspots are handled similarly to receptacles and often overlap receptacles; the hotspot recognizes the mouse events, such as a left mouse button click on a hotspot, cursor enter hotspot or cursor leave a hotspot, and causes logic hunk programs to run. The  
15 appropriate logic hunk programs are created at edit time and are attached by the binder to the hotspot data so the runtime system can execute the flows (logic hunk programs) when needed.

After the above described initialization functions (load data hunk, load logic hunk and build user interface), the runtime engine calls the first logic flow through the logic flow  
20 interpreter, a recursive operation code interpreter. The process followed by the logic flow interpreter is described in **FIG. 19**. The logic flow interpreter parses the logic code and first gets the current program counter (pc) address (an instruction pointer) and gets the op code designated by the pointer. This op code is read to determine whether it is a routine op code which can be executed directly by the interpreter, or a non-routine op code for which the  
25 flow interpreter calls to the operating system or a special routine.

In the preferred embodiment, routine op codes are those which almost any interpreter would execute, such as math and boolean functions and other simple computational functions dealing with the basic handling of production logic, global variables, receptacles and objects. Non-routine op codes deal with specialized functions more unique to the  
30 particular application, such as spatial video logic, loading, hiding and showing visuals, such as windows, viewports and pictures, calling dramatic videos and playing and muting of

sounds such as MIDI and WAV files. Non-routine op codes can also include custom codes created by the developer.

The play dramatic opcode merits additional comment. While it is similar to opcodes dealing with the playing of sound files, it differs in that it requires the spatial video being played to be interrupted. Thus, in the preferred embodiment of the system, the play dramatic opcode allows the runtime system smoothly to incorporate dramatic video with the spatial video. Although there is no requirement that the dramatic video match anything within the spatial video, the overall feel of the environment is improved when there is no visual difference in switching between the two types of video. Similarly to the matching steps described earlier in this document concerning turn and roundabout transitions, the sense of immersion in the environment is enhanced by matching scenes between the transition points in the spatial video and the ends of the dramatic video. The matching process is complicated by that fact that spatial video is typically played back in anamorphic (stretched) form while the dramatic video is played back at the original aspect ratio (unstretched).

Creating the desired matching requires several steps:

1. Choosing the place in the spatial video to transition to the dramatic video.
2. Choosing/creating the appropriate initial frame of the dramatic video which most closely matches the spatial video. Because the dramatic video frame is half the size of the spatial video frame, this matching also includes the offset from the edge that the frame matches.
3. Choosing the place in the spatial video to transition back to when the dramatic video has completed.
4. Choosing the correct end frame of the dramatic video to match the spatial video.
5. Creating a 2 frame clip consisting of the first and last frames of the dramatic video sequence (determined by the previous steps) which has been stretched for anamorphic form.
6. Importing this 2 frame clip into the edit system.
7. Creating a flow which invokes play dramatic with the offset information determined in step 2.
8. Setting a frame exit from the first frame to invoke the created flow.

Once the dramatic sequence has been appropriately wrapped with the matching spatial sequence, it can be treated as any other spatial sequence in the runtime system.

When the runtime system encounters a play dramatic opcode, it first takes the offset information and pans the video frame until it is displaying the picture at the requested offset.

5 It then invokes the MPEG decoder to play the dramatic sequence in the same position as the spatial video. Because the two frames have been matched, to the user there is no apparent change. Once the dramatic video has finished, the spatial video player takes up playing the second frame of the 2 frame clip. As previously discussed, where a hardware player is available, the dramatic video is played by the hardware player in step 7. A flow will also  
10 return to the software decoder once the dramatic video has finished playing.

**FIG. 25** illustrates the relationship between the flow interpreter, the routine op codes and the visual and video op codes. **FIG. 25** is a relationship diagram, not a flow chart. Therefore, while it shows the hierarchical relationship among the components of the runtime system, it does not show all of the recursive connections. It is suggested that the various  
15 flow charts referenced in **FIG. 25** be reviewed for this information. Similar op codes and routines exist for dramatic video sequences, sounds and music (not shown). A list of op codes used in a preferred embodiment is contained in the flow language reference contained in Appendix A and incorporated herein by reference.

In the preferred runtime system illustrated in **FIG. 19**, when the interpreter  
20 encounters a routine op code, the interpreter executes it and updates the position counter and repeats the loop. If the code is a visual load op code the interpreter calls the visual load routine (**FIG. 24**). If the code is a video op code, the interpreter calls the video op code interpreter (**FIG. 20**). The flow interpreter passes to the video interpreter the current video clip ID, the beginning frame number of the current clip, as well as the starting frame number  
25 for the clip and the list of interesting frames (see below). The beginning frame number is the number of the frame at the beginning of the clip, this number is constant for any clip. The starting frame number is the frame number where the video clip is to start playing. This number changes from access to access depending on the user navigational and other input. When the video interpreter exits back to the flow interpreter (see **FIG. 20**) it does so with an  
30 updated position counter value. Other non-routine op codes (not shown) are handled in a fashion similar to that shown for visual op codes.

An 'Interesting' frame is one where something changes other than merely the frame number as the video is played. That is, an interesting frame is one in which the runtime system must do something in addition to simply playing the MPEG video clip. An interesting frame is one where either a hotspot begins or ends, a receptacle begins or ends, a transition/exit to another clip can happen (turn, roundabout or other) or other actions (music, sound) begin or end. Also, the last frame of a clip is 'Interesting'.

The video op code interpreter **FIG. 20** takes the information received from the flow interpreter, determines the next interesting frame in the clip and sets the current frame number to the starting frame number provided by the flow interpreter (see **FIG. 19**). It compares the frame at which it was instructed to begin playing (current frame) and the beginning frame number of the clip. If it finds that the clip is starting somewhere other than the beginning, it calls the **DOEVENTS** routine (**FIG. 22**) in its catch-up mode. Once **DOEVENTS** has caught up, or if the clip is starting at the beginning, the video op interpreter initiates the **DOFRAMES** routine **FIG. 21**. When **DOFRAMES** returns to the video op code interpreter it does so with a new current frame value.

**DOFRAMES** can also return an asynchronous call (see **FIG. 23**). If it has, the video interpreter sends the call to the flow interpreter to call the async routine. If **DOEVENTS** returns an asynchronous **GOTO**, the video interpreter exits to the flow interpreter with a new position counter. If **DOFRAMES** does not return any async call or **GOTO**, the video interpreter calls the **DOEVENTS** routine (see **FIG. 22**) in its normal (catchup = false) mode. **DOEVENTS** may return a synchronous call, if so, the call is sent to the flow interpreter and the video interpreter calls **DOFRAMES** again. If **DOEVENTS** returns an asynchronous **GOTO** the video interpreter exits to the flow interpreter with the new pc. Otherwise, the **DOFRAMES** routine is called yet again.

As shown in **FIG. 21**, the **DOFRAMES** routine is entered with the "stopped" variable set as false. Its first task is to determine whether there are interesting frames which must be executed before more video can be played. If the current frame is an interesting frame the **DOFRAMES** routine exits to the video op code interpreter **FIG. 20**. If there are frames to be played before the next interesting frame, **DOFRAMES** continues until it either reaches the next interesting frame or it determines that the frame is stopped. When the frame is stopped the MPEG player returns a signal that the frame is stopped. This signal is used by the

DOFRAMES routine to make the stopped frame determination. The frame could be stopped because a flow signaled it wants the frame stopped. The frame could also be stopped because the end of the clip has been reached or because the user has paused on a particular frame or has reached the end of the clip without exiting. If the frame is stopped,  
5 DOFRAMES exits back to the video interpreter with a normal return code. This causes the current frame to be repeated and allows the "stopped" frame to be panned and all hotspots, receptacles and the like to be continually checked by the runtime engine.

As shown in **FIG. 21** if a mouse event needs processing or some other asynchronous call or GOTO has occurred DOFRAMES exits with a call or GOTO return code which, as  
10 shown on **FIG. 20**, the video op code interpreter reads and exits to the flow interpreter to call the appropriate routine. If none of these events have occurred, DOFRAMES draws the current frame into the buffer and gets the new current frame number from the MPEG player.

If the new frame number is the same as the previous one, then the current frame is stopped and current frame stopped is set to true, M-views are then drawn into the buffer as indicated  
15 and where indicated and the buffer is copied to screen. The DOFRAMES routine then repeats.

DOFRAMES continuously checks the global variables to determine whether any asynchronous events have occurred (see **FIGS. 23 & 25**). If the DOFRAMES routine detects an asynchronous call (e.g. mouse action) the video interpreter exits to the flow  
20 interpreter where the call routine is performed. After the call routine is processed by the flow interpreter it exits with a return call to the video interpreter and the video interpreter repeats the DOFRAMES routine, this time with its catch-up mode disabled.

If the DOFRAMES routine detects an asynchronous GOTO (such as a transition to another video clip), the video interpreter exits the video clip to the to the new position  
25 counter position. If the DOEVENTS returns a synchronous call, the video interpreter recursively calls the flow interpreter for the call's flow routine to be processed. After the call is finished, or if there was no call, DOEVENTS is called in non-catchup (regular) mode.

As illustrated in **FIG. 20**, the DOEVENTS routine is entered from the video interpreter either in catch-up mode, where a clip is being entered at a position other than the  
30 beginning or in regular mode where there are no interesting frames before the current frame.



The difference in the DOFRAMES modes is that the display and exits are disabled during the catch-up mode. **FIG. 22** illustrates the DOEVENTS routine.

The DOEVENTS routine is entered when the DOFRAMES routine reaches an interesting frame. If the frame number of the next interesting event is greater than the current frame, the DOEVENTS routine determines whether any transition frames remain to be played and, if so, plays them. DOEVENTS then exits to the video interpreter which, in due course, executes the DOFRAMES routine. If the current frame is an interesting frame, the DOEVENTS routine proceeds to process the events which make the frame interesting. These could be sound, music, cache frame, hotspot or receptacle activation or deactivation, transition, exit or clip end. Once the events at the current interesting frame are processed, the routine gets the next interesting frame position from the list. If this frame is not the current frame, the routine exits back to the video interpreter.

Cache Frames and Transition Frames. In the preferred embodiment, the runtime system is set up to interleave the playing of transition frames and the placing cache frames in the cache buffer. Thus, as shown in **FIG. 22** after setting a frame aside in the cache buffer the system instructs the MPEG player to play the next frame of the transition. If all transition frames are finished before all cache frames are loaded, the remaining cache frames are loaded in the "cache frame?" step shown in **FIG. 22**. If there are no more frames to be cached, but further transition frames to be played, the DOEVENTS routine plays the remaining transition frames before continuing with the parsing of the frames on the clip.

In a system that supports overlapped I/O with other computations, such as a multitasking system, the MPEG player could use one thread (or task) to accomplish the seek to the TO clip while the main program continues to play transition frames and any other events which could be executed prior to the availability of the TO clip. If the transition is finished before the TO clip seek is completed, the main program would wait for the seek thread to complete the seek. Once the TO clip is available any remaining transition frames could be played while the cache frames are loading as catch-up mode parses the TO frame. This would allow the transition sequence, which, in the preferred embodiment, does not require the TO clip, to mask seek time to the TO frame. As described more completely in the Special Effects Applications, the transition frames can be either generated on the fly or kept in memory, so no disk access is required to play them.

Hotspot and Receptacle Activation. Hotspots and Receptacles are activated when they become visible and deactivated when they become invisible by DOEVENTS adding and removing them from the active hotspot or active receptacle list. Adding them to the active list makes the frame-by-frame size and position information available to the MPEG player and the asynchronous event handler (see FIG. 23).

Sound Events. Events that begin music or sound call upon the appropriate operating system service.

End of clip events. End of clip events require the DOEVENTS routine to back up one frame. This continues to run the video and logic to allow the user to navigate or to execute any available exits or other actions.

Cache Frame events. Cache frame events tell the runtime system which, in turn, instructs the MPEG streamer to store a given frame of video rather than send it to the MPEG player for decoding. In the preferred embodiment at bind time all of the frames to be cached for any transitions possible from a particular clip are physically located at the beginning of that clip and cache frame events are generated to load them when the clip is accessed. These transition frames are preferably reference frames. If MPEG coding is used, the frames are preferably I frames. At runtime, when the clip is accessed the MPEG streamer loads the cached MPEG frames into a cache frame buffer. The MPEG streamer sends the proper cached frame to the MPEG player future buffer at the beginning of the transition sequence. The MPEG streamer then sends the proper set of transition B frames, as described in the Special Effects Applications, to the decoder. The use of the cached TO frames and the transition B frame sequence, which can also be cached, in transitions from one clip to another allows the immediate construction of transition frames with both FROM and TO frame information for a meaningful transition. This can occur while the system is loading the cache frames for the TO clip or, if a multitasking operating system is used, while the system is seeking the TO clip or frame. Where a clip is entered from a previous clip through a transition the caching of frames can be interleaved with the playing of the transition sequence.

When the transition is being interleaved with the loading of cached frames from the TO clip the DOEVENTS routine instructs the MPEG streamer to play the next frame of the transition sequence before completing the loop to get the next interesting frame, which may

be another cache frame. In the preferred embodiment, the transition frames are assigned one or more invalid frame numbers, such as negative numbers. Thus, when the MPEG player provides the system with the frame number it does not increment the current frame number which, consequently, remains unchanged throughout the transition. As the current frame number does not change, DOEVENTS (**FIG. 22**) does not return to video until all cache frames are stored and all transition frames are shown. This allows the loading of the cache frames for the TO clip into the streamer's cache frame buffer to be interleaved with the decoding and playing of the transition sequence frames without a loading delay after completion of the transition.

Transition events. Transition events which make smooth transitions from one clip to another by using a previously cached frame to hide the time spent seeking to and preparing to display the next clip. They allow smooth transitions of various types as described elsewhere herein and in the copending Special Effects Applications.

Clip Exit events. Clip Exit events return control to the flow interpreter which allows the interpreter to cause various designated logic hunk programs to run. These programs can play sounds, set global state variables, keep score, and the like. By changing global state variables, the programs associated with a Clip Exit can change the currently playing video clip to another clip. An associated logic hunk program can also test global state variables to decide what to do.

Further, the Clip Exit events only call the associated logic hunk program if a particular global state variable, EXIT\_ID, matches the Clip Exit's ID. See **FIG. 22**. There are a number of possible values for the Clip Exit ID: Always Match, Left Turn Match, Right Turn Match, User Data Match. A Clip Exit with an ID of Always Match will always call the logic hunk program. A Clip Exit with an ID of Left or Right Turn Match will only call the logic hunk program if the runtime system has detected the user is trying to turn left or right. A Clip Exit with an ID of User Data Match will only call the logic hunk program if the global user data match variable matches the Clip Exit ID's value. When a Clip Exit is taken it may Call or GOTO the logic hunk program. A Call will leave the current video playing; a GOTO will terminate the current video.

The method in which the flow and video interpreters of the runtime system receive asynchronous inputs is through use of a global variable table, a method well established in

the art. The runtime system receives user input through an asynchronous event handler **FIG. 23** which detects user actions and communicates with the runtime system through a global variable list (see **FIG. 25**). The asynchronous event handler updates the global variable list in response to mouse events or other user input in the method illustrated in **FIG. 23**. Further, the asynchronous event handler checks global variables regarding hotspot location and activation established by the flow interpreter. As shown in **FIG. 23**, the asynchronous events handler detects cursor moves created by a mouse (or any other input device) as well as clicks. When the asynchronous events handler detects a mouse move, it first sends the new mouse coordinates to the MPEG player panning control so that the screen can be adjusted accordingly. If the screen is panned to the maximum in the direction of the mouse move, the handler sets a global exit variable for the proper transition. In addition, if the mouse moves or if it is clicked, the handler checks the list of hotspots, receptacles and visuals that are active and determines whether the coordinates and mouse events it has detected require performance of a hotspot, receptacle or visual associated call or GOTO. If so, the async event handler sets an async call or GOTO in the global variable list and returns to the operating system. The global variable list is checked by the flow interpreter and depending on the nature of the event, it is executed by the DOFRAMES or DOEVENTS routine (see **FIG. 20**).

Visuals are handled by the flow interpreter **FIG. 19** by calling the visual load routine **FIG. 24**. The visual load routine simply gets the relevant information about the visual from the visual table, creates the window needed and calls any flow routine associated with the visual. The visuals are rectangular areas of the display, sometimes called windows, in which data, in the form of bitmaps, video or a viewport are displayed. Visuals can be placed within each other. Thus one can have a parent window and a child window. Hence, the visual load routine is called recursively to detect and load any child windows and run their associated flows. For example, a window visual may have a viewport visual playing video within it.

Navigation is handled through the asynchronous events handler. **FIG. 23**. In the preferred embodiment the screen is divided into seven regions for the purposes of navigation.

These regions are illustrated in **FIG. 27**. Region 0 is in the center; while the cursor is in region 0 the user is not trying to turn or change viewing direction. If the cursor is in regions 1 or 4 the asynchronous events handler provides these coordinates to the global variables

table and the flow interpreter initiates a flow which directs the MPEG player to pan the MPEG frames one pixel per frame to the left (in case of 1) or right (in the case of 4). In regions 2 or 5 the instruction is to pan at the rate of 4 pixels per frame. In regions 3 or 6 the direction is to pan by 16 pixels per frame. When the image is panned to the maximum to the left or right and the cursor remains in region 3 or 6, the global variable ExitID is set to Left Turn Match or Right Turn Match. If the cursor remains in the farthest left or right region (regions 3 or 6) when a frame for which a left or right turn transition has been constructed, if the exitIDs match the DOEVENTS routine calls the transition routine which is paired with the appropriate clip exit and the newly entered clip is started. See FIG. 22.

While the various navigation regions could be considered region hotspots, they are preferably recognized by the asynchronous events handler FIG. 23 which places the appropriate panning instruction in the global variable list. Forward and reverse navigation can be handled in any logical manner. For example, there could be regions on the top and bottom of the screen which, when detected by the asynchronous events handler could set a global variable for forward or reverse play. Preferably, however, the forward speed controlled by dragging the mouse rather than by its position on the screen. Forward speed is sensitive to a left button mouse drag toward the top of the display. The degree of increase of speed is proportional to the length of the drag. A left button mouse drag toward the bottom of the display proportionately slows or stops the video play. If the production allows reverse play of the MPEG video clip (see Special Effects Applications), this could also be accomplished by a further mouse drag toward the bottom of the display when forward motion is stopped. However, reverse play is not required and, because of the availability of the roundabout maneuver of the invention, may be omitted without significant loss of the sense of being immersed in the video environment.

FIG. 25 provides an overview of the runtime system showing the relationship between the flow interpreter, the video interpreter and the various routines as well as the interaction between the asynchronous event handler and the flow interpreter through the global variable table.

FIG. 26, is a table which contains excerpts from the code played along with the MPEG video stream. The first column is the code line number, the second column is the source line which, when the system is playing a video clip will contain the clip's frame

number. Code line 5f is associated with the display of frame 0. Code line 73, frame 20 is a hotspot activate event. The IDX:27 is an index to the hotspot table contained in the Data Hunk of the bound production. This table tells the system where the hotspot is in the following frames, until line 2a3 when the hotspot is deactivated. The hotspot was activated  
5 in frame 20 either because it came onto the screen at that frame, or the user was close enough to it to see it, or simply because the person editing the production decided that the hotspot should be activated at this place. A hotspot is deactivated for similar kinds of reasons.

Playing the video clip. As shown in step two above once the interesting clip file has been parsed to determine the next interesting clip the video stream is played in the following  
10 manner. Each frame of video is decoded from the MPEG stream into a buffer. After decoding the buffer holds a bitmap of the image that is about to be displayed. Before the image is displayed any pending events (step 3 above) are handled. Then any active receptacles are handled. For each active receptacle the video interpreter determines if there is an M-view in the receptacle. If so, the interpreter determines the receptacle's position and  
15 size and angle from the active receptacle list. It then renders the M-view into the buffer holding the bitmap of the image to be displayed. In rendering the appropriate M-view bitmap, the transparent parts of the M-view are not drawn into the buffer, only the opaque parts are.

The DOFRAMES routine draws the M-view into the display buffer, as shown in  
20 **FIG. 21.** The display buffer contains a frame which is to be displayed. The frame may already have other M-views rendered on it. The M-view associated with the active receptacle is drawn by using the active receptacle list and the receptacle by frame list which tracks receptacle size and angle by frame number. The particular M-view associated with the receptacle is also present as an object property variable. As discussed above, the variable is  
25 subject to change by flows responsive to game logic. The M-view draw routine looks to this variable to retrieve the appropriate M-view. The M-view's resolution and angle are retrieved from the M-view directory using the frame specific information in the receptacle by frame list.

The following steps allow for the rendering of an M-view onto a buffered bitmap.  
30 Given a buffer to render into (which buffer already contains a frame of video, and may already contain other previously rendered M-views), and given the position and size, as well

as the M-view scale and angle. In the preferred embodiment the appropriate angle has already been determined at edit time by the developer using the M-view editor. At bind time, as previously discussed, the interpolated sizes, positions and angles are converted to frame specific sizes, coordinates and angles and placed in appropriate tables in the Data Hunk. During runtime, the appropriate M-view face is stretched or shrunk and is copied to the buffer at the correct location of the video frame. The M-view rendering program decodes the run length compressed, bit-packed bytes, and color look up table associated with the M-view, which were created and placed in the production at Bind Time and performs the following steps:

10        STEP ONE: Normalize the angle. As the interpolator may end up with a degree number outside this range, but all angles in the M-view angle index are between 0 and 359 degrees the first step is to normalize the angle associated with the receptacle. This is done by adding or subtracting 360 degrees to any angle not within the range. This step can be done on the fly at runtime or can be done by the binder and the appropriate angle entered in to a  
15        table in the runtime Data Hunk.

STEP TWO: Determine the scale. Next the appropriate scale for the M-view face to be overlaid is determined by multiplying the size supplied by the receptacle with the size stored with the M-view. This allows the M-view of objects with vastly different absolute sizes to be associated with the same receptacle. For instance, a user may wish to put either a  
20        fly or an elephant into a particular receptacle. The receptacle's size need only be set once at edit time to make either the fly or the elephant look correct in the scene. The other will be scaled accordingly.

STEP THREE: Determine the position. Receptacles, like hotspots, are rectangular. They have an origin or anchor points at one of nine positions: top-left, top-center, top-right, middle-left, middle-center, middle-right, bottom-left, bottom-center, bottom-right. The  
25        origin that is assigned at edit time remains constant regardless of the scale computed in Step Two. Depending on the origin/anchor point chosen, the location of the bitmap of M-view rendered in the receptacle varies. If the receptacle origin is bottom center, the center of the bottom edge of the M-view bitmap will be on the origin. The M-view rendering program  
30        uses the supplied position, origin, and scale to determine the size and location of the rectangular M-view bitmap which is to be displayed.

STEP FOUR: Stretch or shrink M-view face. In order that the M-view bitmap be the right size it may have to be stretched or shrunk. This is done to the bitmap corresponding to the M-view face that most closely corresponds to the normalized angle from Step One to fit it into the receptacle from Step Three. The transparent pixels from the M-view are not copied into the buffer. Stretching or Shrinking can use any of a number of well known algorithms. In the preferred embodiment we use a very fast algorithm that sacrifices some quality of final image for speed and takes advantage of run length encoding of the transparency information. Computer Graphics: Principles and Practice by Foley & van Dam describes several algorithms that do sophisticated stretch/shrink manipulations of bitmaps to make the destination pixels blend to be close to the color of the source. However, in the interest of computational efficiency, we have chosen an algorithm which does not blend colors. Approximately evenly spaced rows and columns are simply duplicated when expanding the bitmap or simply dropped when shrinking. When scaling up some source (original) pixels go to more than one destination pixel (the adjacent row or column). When scaling down some source pixels are not used in the destination. We use either binary fraction or Bresenham's digital differential analyzer, which is set out in Foley & van Dam, p. 74, to keep track of fractional pixels.

Runtime Logic. The spatial video production logic can also be illustrated and better understood by looking at examples of disassembled code from a production made with a preferred embodiment of the system. Such examples are contained in Tables 1 through 5 which are attached to and incorporated in this specification. **Table 1** illustrates the code used in a preferred embodiment of a spatial video production to load up the user interface. The 8 digit numbers on the left margin are simply offset numbers in the logic hunk. They were added by the debugger software used to disassemble this portion of the code. This section of code is executed by the flow interpreter.

**Table 2** may provide additional appreciation for how logic execution and video frame playing run in parallel in spatial video. Again the 8 digit numbers in the column on the left margin are logic hunk offsets. The second column is the frame number, containing the abbreviation FRAM: followed by the frame number. The column after the frame number column shows the interesting events associated with the frame number. For example, line 73, frame 20 is a HOTSPOT activate event. The IDX:27 is an index to the hotspot table



produced by the binder in the Data Hunk. The hotspot table gives the location of the hotspot in the frames of the clip. Line 2a3, frame 318 shows that the hotspot is deactivated. Reasons for the activation of a hotspot vary. The decision is made by the developer at edit time. The developer could choose to activate a hotspot in a particular frame because the visual feature with which the hotspot is associated became visible in that frame, or because the feature became large enough to distinguish, or for other reasons. Activation of a hotspot could be associated with a flow. That is, the hotspot is not activated until the player succeeds in locating certain objects or otherwise meeting certain requirements of the game logic necessary to activate the hotspot.

The FRAME\_POSITION listing shows the X, Y and Z coordinates of the camera for that frame. These coordinates are useful during edit time to construct maps and the like, as previously discussed. The coordinates can also be useful at runtime. For example, if a map of the environment visual is depicted the OCX which is used to show the player his position in the environment on the map visual may get frame position data to locate the pointer on the map. Further, where there are noisy objects in the environment, the level of the noise can be adjusted by a OCX which adjusts the level in the sound card using the position of the noise source and the FRAME\_POSITION data as the player approaches or retreats from the source.

Notice that the listing ends with a CLIP\_END. This interesting frame instructs the MPEG player to back up one frame and repeat. See FIG. 22.

Table 3 shows more code in a video clip logic section of a production made with a preferred embodiment of the invention. The code in this table is at the start of a particular source video clip named "24 EastAcrossFront." Notice that the first events to occur are to cache destination or "TO" frames used in various transitions possible off of this video clip. This code section also shows a receptacle activate on frame 741, hotspot activations on frames 744 and 746. Notice, as previously discussed, that the RUNTIME\_TRANSITION is always paired with and immediately precedes CLIP\_EXIT, as a turn or roundabout transition always moves the player to another clip. In contrast, CLIP\_EXITS are not always associated with transitions. They may appear without a transition and, by way of illustration, may call a flow which causes a dramatic video to play, exits the game, or accesses another clip at its beginning without a smooth transition event.

**Table 4a** shows a flow in a production made using a preferred embodiment of the method of the present invention. The flow was typed by the developer in the logic editor portion of the spatial video editor. This particular flow is assigned to a hotspot's click event.

The hotspot to which the flow is assigned appears over a poster of a baseball player in the spatial video clip. The flow causes a picture of the baseball player to be displayed and then waits for the player to click. Once the click occurs the flow removes the picture and returns to the video. Notice that the flow is called by the player of the production and is interpreted by the flow interpreter portion of the runtime system. **Table 4b** shows the actual code the binder generated from the above flow.

**Table 5** is a disassembly of part of the hotspot data from the Data Hunk of a production made with a preferred embodiment of the method of the invention. The disassembler used to create this table did not create data file offsets, but, of course, they exist. However, the difference in the disassemblers used for the logic hunk and the data hunk account for the lack of the first offset number column in **Table 5**. The first listings in **Table 5**, those up to the first double space, show a directory of hotspot lists. The list includes information on region hotspots as well as spatial video hotspots. Notice that the offset number given for the region hotspot, 3890, is the same as that for the clip (video tracking) hotspot. This is because this particular production contained no region hotspots. This is confirmed by the listing after the double space showing the count of region hotspots as 0.

After the second double space, the listing goes on to provide information regarding the clip hotspots in the production, providing offset location information on the first occurrence of the hotspot, the flow associated with the hotspot, the ExitID for the hotspot which goes to the global variable table if the hotspot is triggered and the type of trigger which activates the hotspot (e.g., mouse move, mouse click, etc.) and the way the cursor is depicted when it is over the hotspot. This information is followed by a series of listings providing the size and location of the hotspot on the various frames where it appears. These figures are calculated by the binder using the same extrapolation/interpolation methodology used at edit time. Only the first and last positions are given. Ninety-two are omitted from the table in the interest of brevity. The table proceeds to show other hotspot occurrences on various clips of the production.

While the interactive multimedia process and apparatus herein described constitute the preferred embodiments of the present invention, it is to be understood that the invention is not limited to this precise form and that changes may be made therein without departing from the scope of the invention which is defined in the appended claims.

APPENDIX A

## FLOW LANGUAGE REFERENCE

Flows are used in Video Reality productions to allow the product designer (VRED user) to specify execution logic for the product. A flow may say what happens when the product user clicks on hotspot X, or when the video player gets to the end of clip Y. The logic in a flow is specified in a simple programming language that may be entered and changed in VRED. Once entered, the flow logic will be scanned by VRED for obvious errors or omissions. Further error checking will be done by the VR binder, when the production executable is generated. At that time the flow logic will be translated into its final form to allow efficient playback.

All flows start with a FLOW statement and end with an ENDFLOW statement. Between these are a list of executable statements like assignments, loops, conditionals, etc. Every executable statement starts with a keyword that identifies what kind of statement it is. These keywords are:

**assign**

- **call**
- **do**
- **exit**
- **for**
- **if**

- **jump**
- **load**
- **midi**

**return****select**

- **set**
- **setcursor**
- **visual**
- **waitforuser**

**while**

**Comments:** As in Basic, the apostrophe is used to begin a comment. The comment terminates at the next end-of-line.

**Identifiers:** Names for variables, variables, etc. are sequences of alpha characters, digits, and underscores. Names must begin with an alphabetic character (a-z) or underscore (\_). Characters after the first may be alphabetic (a-z), numeric (0-9), or underscore (\_). The maximum length for a name is 64 characters.

**Database Names:** The flow compiler recognizes names for many database objects. Names used for these objects must all be unique - for example, you can't have a bitmap and a clip exit with the same name - and each will be translated to the correct type of reference automatically by the flow compiler. You cannot refer to the name of an object that you haven't added to the database yet!

**Data Types:** The flow language recognizes four basic data types:

Numeric - a four-byte integer

- Boolean - a TRUE/FALSE value
- String - character string up to 255 bytes

Memo - character string of any length. Includes a two-byte length code before the string.

Variables of these types may be declared via the Global Properties Editor in VRED. Note that all database objects (like receptacles, bitmaps, clip exits, and so forth) are represented inside the flow

language as four-byte integer values. This is not ideal, and a future version of the language will recognize more types to handle this problem.

5 **Builtin global variables:** All builtin variables are read/write numerics. They can be used without declaring them or defining them in the database.

10 **QueuedExitID** - Sets up the next exit. Used when the user explicitly clicks on a hotspot associated with an exit, or when the user indicates a general direction like "Take the next right". You can assign a specific clip or flow exit by name, or you can assign one of the exit constants **ExitLeft**, **ExitRight**, or **ExitDefault**.

**ShowHotspots** - If nonzero, causes a box to be drawn around all active hotspots on the screen.

**ShowReceptacles** - If nonzero, causes a box to be drawn around all active receptacles on the screen.

15 **Speed** - Sets the navigation speed in terms of percentage of normal speed. 100 is normal speed; 50 is a slow speed; 200 is a fast speed. Don't bother setting it above 200 or below 0.

**Acceleration** - Sets the maximum rate of change of speed per frame. Not as useful as we thought it would be. Set it to 9999 and leave it.

20 **Xpos, Ypos, Zpos** - Current camera X, Y, and Z position in millimeters relative to an arbitrary (0,0) point. Only available if the FRAME\_POSITION option was used to generate the production. Read only.

**Bearing, Pitch, Roll** - Current camera bearing, pitch, and roll, available if FRAME\_POSITION option is selected. Read only.

**Act** - current act. Initialized to 0 when the production starts, should be maintained by your flows if you care about its value.

25 **Room** - current room. Set each time a new clip is jumped to or called if that clip was associated with a room in the database. Read only.

**Clip** - Database ID of the current clip. Can be used for comparison purposes only. Read only.

**Builtin constants:** The following items are read-only constants.

30 **True** - Boolean constant

- **False** - Boolean constant

- **Random** - Returns a random integer from 0 to the largest four-byte integer.

- **ExitLeft** - Indicates a left turn; use with **QueuedExitID**.

- **ExitRight** - Indicates a right turn; use with **QueuedExitID**.

35 • **ExitDefault** - Indicates that no turn should be taken; use in conjunction with **QueuedExitID**.

- **Nothing** - Sets variables that deal with database objects so that they don't refer to any DB object.

**Builtin variables of objects, receptacles, and visuals:** All builtin object variables are read/write numerics unless otherwise noted.

40 Receptacles have a builtin variable called **CONTENTS** variable of type numeric. You can set the CONTENTS field to an MVIEW to cause the specified MVIEW to show up in the receptacle.

Receptacles and Hotspots both have builtin variables called **ENABLED** which determine if that object is enabled. Objects which are not enabled do not appear on the screen and cannot be interacted with by the user.

45 Custom OCX visuals have whatever variables their creator built them with.

Picture visuals have a numeric **CONTENTS** variable. Assign a bitmap to this variable to cause that bitmap to show up in the picture visual.

**Operators in Expressions:** The following operators are supported (highest to lowest precedence):

50 - (unary negation)

Not

- \* / MOD

- + - &

- < > >= <= = <>

55 • AND

- OR

## FLOW LANGUAGE STATEMENTS

In the following flow statements, italics are used to represent arbitrary names or expressions that should be invented by the person editing the flow. These words have no special meaning to the flow compiler. Items in square brackets ( [ and ] ) are optional and may or may not be present in a real flow. In either event, the square brackets are not present when you write the flow. Flow language keywords are represented in bold face.

**assign** *name* = *expression*

Assigns the value of the expression to the variable with the given name. This is a synonym for the **set** command.

**call dramatic** *name* [*start\_ulx start\_uly start\_width start\_height* [*end\_ulx end\_uly end\_width end\_height*]]

Plays the specific dramatic video, then returns to the flow. When the **call dramatic** statement finishes, the dramatic video is over. The flow can force spatial video to pan to a desired location before playing a dramatic video. It can also force the spatial video that follows a dramatic video to begin at a pre-defined pan location. This is done by adding up to eight optional parameters to the CALL DRAMATIC statement. The eight parameters define two rectangles in an (x, y, width, height) format. The runtime system will pan the video to the area described by the first rectangle before playing the dramatic video, and will reset the pan point to the second rectangle after playing the dramatic. Each X value should be in the range 0 - 351 (the width in pixels of standard MPEG), each Y value should be in the range 0-239, each Width value should be 1-352, and each Height value 1-240. The default values are (88, 0, 176, 240) for both rectangles, giving the center portion of the source video. Also see **load dramatic**.

**call flow** *flowname*

Executes the specified flow and returns.

**do**

*statement-list*

**until** *expression*

Executes the statement or statements in *statement-list* until the *expression* is not true. The statements in the *statement-list* will be executed at least once.

**endflow**

Terminates the flow and returns to the flow's parent.

**flow** *flowname*

Begins a new flow with the name 'myflow'.

**for** *variable* = *start-expression* **to** *end-expression* [**step** *step-expression*]

*statement-list*

**loop**

Maintains a counter variable *variable* and execute *statement-list* while the counter variable is less than the specified ending expression. If no *step-expression* is given, the counter variable is incremented by one each time through the loop. The **for** loop is equivalent to the following **while** loop:

*variable* = *start-expression*

**while** *variable* < *end-expression*

*statement-list*

**set** *variable* = *variable* + *step-expression*

**loop**

**if** *expression* **then**  
     *true-statement-list*

5   **[else**  
     *false-statement-list*]

**endif**

10         Evaluates the expression, and if it is nonzero, executes the statement or statements in the *true-statement-list*. If the **else** part is present and the expression is false, the statement or statements in the *false-statement-list* are executed.

**jump flow** *flow\_name*

15         Executes the specified flow and returns to the parent of the flow containing the **jump flow** instruction. The parent of the flow containing the **jump flow** instruction becomes the parent of the called flow.

**jump video** *video\_name*

20         Plays the specified spatial video and returns to the parent of the flow containing the **jump video** instruction. The parent of the flow containing the **jump video** instruction becomes the parent of the called flow.

**load dramatic** *name*

25         Preloads the given video for use as a dramatic clip. The runtime system may be able to provide faster switchover from spatial to dramatic video if given some advance notice that the dramatic will be required. Use a **load dramatic** statement from a flow to give that notice. Also see **call dramatic**.

**midi balance** *slot balance*

30         Sets the left-right speaker balance of the MIDI file playing in the given slot. The *balance* parameter indicates the desired left-right balance, with +100 indicating 100% on the right and -100 indicating 100% on the left.

**midi mute** *slot*

35         Mutes the MIDI file playing in the given slot.

**midi play** *midi\_name slot looping*

40         Plays the specified MIDI file. The *slot* parameter specifies the MIDI slot to use for the file. 0 and 1 are valid MIDI slots. The *looping* parameter specifies whether to start the MIDI clip over again when it ends or terminate it. Some spatial video clips also have MIDI files attached to them that automatically begin playing when the clip is played.

**midi stop** *slot*

Terminates the MIDI file playing in the given slot.

45   **midi repeat** *slot repeat*

Changes the value of the *looping* parameter chosen when the **midi play** command was issued.

**midi unmute** *slot*

50         Unmutes the MIDI file playing in the given slot.

**midi volume** *slot level*

Sets the volume of the MIDI file playing in the given slot. 0 is minimum volume and 100 is maximum.

55

**select case** *select-expression*

```

case constant-expression-1
    statement-list-1
[
    case constant-expression-2
        statement-list-2

```

```

    ...

```

```

]
[
    case else
        statement-list-else
]

```

```

endselect

```

The **select** statement is a way to choose different statement lists to execute based on the value of the *select-expression*. The *select-expression* must be of type numeric.

```

setcursor defaultcursor = cursor-name

```

```

setcursor heldcursor = cursor-name

```

Sets the default or held mouse cursor to the cursor matching the specified name. The default mouse cursor is used when the mouse cursor is not over a hotspot or active receptacle; the held cursor is used when the mouse is over a hotspot or active receptacle.

```

visual hide visual_name

```

```

visual show visual_name

```

```

visual load visual_name

```

```

visual unload visual_name

```

Performs operations on visuals. There are four major types of visuals: **pictures**, **viewports**, **windows**, and **customs**. Picture visuals contain bitmaps. Viewport visuals contain video. Window visuals contain other visuals. Custom visuals are OLE controls and are managed by an external program.

The various visual operations allow you to perform operations on visuals. The **load** operation causes the visual to be loaded off disk. **Unload** is the opposite. The **show** operation causes the visual to be displayed. **Hide** is the opposite of that. The VRED visual editor allows you to choose whether the visual is automatically loaded and/or shown when the production starts up.

```

while expression
    statement-list

```

```

loop

```

Executes the statement or statements in *statement-list* while the *expression* is true. If the expression is initially false, the statements in the *statement-list* will not be executed.



## EXAMPLE FLOWS

Play a wav file:

```

5      flow PlayWav
        call wav wav_name
      end flow

```

Play dramatic video:

```

10     flow PlayDramatic
        call dramatic dramatic_name
      end flow

```

Play dramatic video only the first time we get to a location:

15        This flow requires an integer global variable (property). Use the VRED property editor.

```

        flow PlayDramatic
          if (integer_global_variable = 0) then
            call dramatic dramatic_name
20         set integer_global_variable = 1
          end if
        end flow

```

Display a picture in a picture visual, wait for user to hit a key:

```

25     flow ShowPicture
        set picture_visual.contents = bitmap_name
        visual show picture_visual
        waitforuser
30     visual hide picture_visual
      end flow

```

Play one of five random wav files:

```

35     flow PlayRandomWav
        select case (random mod 5)
          case (0)
            call wav first_wav
          case (1)
40         call wav second_wav
          case (2)
            call wav third_wav
          case (3)
            call wav fourth_wav
          case (4)
45         call wav fifth_wav
        endselect
      endflow

```

## FLOW LANGUAGE GRAMMAR

Nonterminals start with CAPITAL

Terminals in lower case

Keywords and operators in "quotes"

5

The grammar does not spell out the specific relationships needed to deal with these expressions, but since most people know how they "should" work, it's OK. Parentheses are allowed to group expressions. Keep in mind that any "operand" may be an expression.

10

Starting nonterminal is Flow

Flow            $\leftarrow$  "flow" FlowIdent Stmtlist Endflow

15

FlowIdent       $\leftarrow$  identifier

$\leftarrow$  null

Endflow        $\leftarrow$  "end" "flow"

$\leftarrow$  "endflow"

20

Stmtlist        $\leftarrow$  Stmt Stmtlist

$\leftarrow$  null

Stmt            $\leftarrow$  "if" Conditional

$\leftarrow$  "do" Doloop

25

$\leftarrow$  "while" Whileloop

$\leftarrow$  "for" Forloop

$\leftarrow$  "select" Casestmt

$\leftarrow$  "exit"

$\leftarrow$  "set" Assignment

30

$\leftarrow$  "assign" Assignment

$\leftarrow$  "setcursor" SetcursorAssignment

$\leftarrow$  "call" Call

$\leftarrow$  "jump" Jump

$\leftarrow$  "load" Load

35

$\leftarrow$  "visual" Visualstmt

$\leftarrow$  "midi" Midistmt

$\leftarrow$  "move" Movestmt

Conditional    $\leftarrow$  Operand "then" Stmtlist Elseclause

40

Elseclause      $\leftarrow$  "else" Stmtlist Endif

$\leftarrow$  Endif

Endif           $\leftarrow$  "end" "if"

45

$\leftarrow$  "endif"

Doloop          $\leftarrow$  Stmtlist "until" Operand

Whileloop       $\leftarrow$  Boolexp Stmtlist "loop"

50

Forloop         $\leftarrow$  identifier "=" Operand "to" Operand Stepclause Stmtlist "loop"

Stepclause      $\leftarrow$  "step" Operand

$\leftarrow$  null

55

	Casestmt	← "select" Operand Caselist	
	Caselist	← "case" Caselist2 ← "end" "select"	
5	Caselist2	← Operand ← "(" Operand Testval ")" Stmtlist Caselist ← "else" Stmtlist	
10	Testval	← "," Operand Testval ← "to" Operand Testval ← null	
15	Assignment	← identifier "=" Operand	<i>expression</i>
	SetCursorAssignment	← identifier "=" Operand	<i>zero or name of cursor entry in Library. Identifier must be one of: DefaultCursor, HeldCursor, HeldHotspotCursor</i>
20	Call	← Jumptype ← Loadtype	
	Jump	← Jumptype	
	Load	← Loadtype	
25	Jumptype	← "video" token ← "flow" token	<i>name or id of a video name or id of a flow</i>
30	Loadtype	← "wav" Operand ← "dramatic" Operand	<i>name of a WAV file name of a dramatic video file</i>
35	Visualstmt	← "hide" token ← "show" token ← "load" token ← "unload" token ← "switch" token	<i>name or id of a visual name or id of a visual name or id of a visual name or id of a visual name or id of a visual</i>
40	Midistmt	← "play" token,token,token,token ← "stop" token ← "continue" token ← "mute" token ← "unmute" token ← "volume" Operand Operand ← "balance" Operand Operand ← "repeat" Operand Operand	<i>name or id of a midi name or id of a midi name or id of a midi name or id of a midi name or id of a midi name or id of a midi name or id of a midi</i>
45	Movestmt	← identifier "to" identifier	
50	Operand	← integer ← string ← identifier ← "(" series_of_tokens_with_balanced_parentheses ")"	

## LOW-LEVEL FLOW LANGUAGE OPCODES

The statements in the flow language are compiled using standard techniques in the art of compiler and interpreter design into a target language for a virtual computer. The virtual computer's machine language is described below.

The virtual machine is assumed to have 32 registers of each of the four basic operand types: numeric (four-byte integers), boolean (yes/no values), string (fixed 255-byte character strings) and memo(variable-length character strings with a two-byte length prefix.) All operands in the tables below are four-byte integers unless otherwise specified.

In the low-level flow language, runtime system objects like receptacles, visuals, MIDI files and so forth are often referred to by means of an index number. The binder assigns these index numbers. They are transmitted to the runtime system either via the initialization file that it creates, in the case of external assets like MIDI files and dramatic video files, or inside the data hunk, in the case of receptacles and visuals.

opcode	operands	Meaning
opSourceLine	Length, text (string of 'length' characters)	Used for debugging; text string contains the text of the line of flow code that generated the following instructions
opCallLogicBlock	Logic offset	Push current location on the call stack and branch to specified offset in the logic section. Continue executing instructions from that point.
opReturn		Pop the call stack and branch to the location specified. Continue executing instructions from that point.
opGoto	Logic offset	Jump to the specified offset in the logic hunk and continue executing instructions.
opFlowLogicStart		Signals the beginning of a flow
opVideoLogicStart	Start_frame, return_label	Signals the beginning of a video. Start_frame is the frame number of the first frame of the video in the clip hunk; return_label is the offset in the logic hunk of the opReturn instruction for this video. Between the end of this instruction and the return_label offset is a series of frame events for the target video stream.
opExitProduction		Terminates the entire production.
opHideWin	Window_index	Causes the specified window visual to be hidden, but not unloaded.
opHideViewport	Viewport_index	Causes the specified viewport visual to be hidden, but not unloaded.
opHidePicture	Picture_index	Causes the specified picture visual to be hidden, but not unloaded.
opHideCustom	Custom_index	Causes the specified user-written OCX to be hidden, but not unloaded.
opShowWin	Window_index	Causes the specified window visual to be displayed.
opShowViewport	Viewport_index	Causes the specified viewport visual to be displayed.
opShowPicture	Picture_index	Causes the specified picture visual to be displayed.
opShowCustom	Custom_index	Causes the specified user-written OCX to be displayed.
opLoadWin	Window_index	Causes the specified window visual to be loaded, but not displayed.
opLoadViewport	Viewport_index	Causes the specified viewport visual to be loaded, but not displayed.
opLoadPicture	Picture_index	Causes the specified picture visual to be loaded, but

opLoadCustom	Custom_index	not displayed. Causes the specified user-written OCX to be loaded, but not displayed.
opUnloadWin	Window_index	Causes the specified window visual to be unloaded
opUnloadViewport	Viewport_index	Causes the specified viewport visual to be unloaded
opUnloadPicture	Picture_index	Causes the specified picture visual to be unloaded
opUnloadCustom	Custom_index	Causes the specified user-written OCX to be unloaded
opPlayMidi	Midi_index, slot, looping (boolean)	Plays the specified MIDI file in the given MIDI slot. If LOOPING is true, the file starts over when it is finished. Two MIDI slots are available, zero and one.
opStopMidi	Slot	Stops the MIDI file playing in the specified slot.
opMuteMidi	Slot	Mutes the MIDI file playing in the specified slot.
opUnmuteMidi	Slot	Unmutes the MIDI file playing in the specified slot.
opSetMidiVolume	Slot, volume	Sets the volume of the MIDI file. Volume is a scale of 0 to 100.
opSetMidiBalance	Slot, balance	Sets the stereo balance of the MIDI file. 0 is equal, positive numbers bias towards the right, negative towards the left. 100 is max right and negative 100 is max left.
opSetMidiLooping	Slot, looping (boolean)	Changes the LOOPING value of the specified MIDI file
opLoadReceptacle	Register, ReceptacleIndex, Offset	Loads the value at offset 'Offset' in the specified receptacle into the specified register.
opStoreReceptacle	Register1, ReceptacleIndex, Offset	Stores the value in 'Register' into the receptacle specified at offset 'Offset'.
opLoadGlobal	Register, Offset	Loads the value at offset 'Offset' in the list of built-in variables into the register specified by 'Register'.
opStoreGlobal	Register, Offset	Stores the value in register 'Register' into the offset in the built-in variable structure that is specified by 'Offset'
opRandom	Register	Generates a random number and puts it in the numeric register 'Register'
opLoadVisual	Register, VisualType, VisualIndex, Offset	Loads the value at offset 'Offset' in the visual identified by VisualType and VisualIndex into the register 'Register'
opStoreVisual	Register1, VisualType, VisualIndex, Offset	Stores the value in 'Register' into the specified visual at offset 'Offset'
opCallDramatic	Register	Calls the dramatic video file whose index is in the numeric register 'Register'
opLoadDramatic	Register	Loads the dramatic video file whose index is in the register 'Register', but does not call it. This can be used to prepare a dramatic file ahead of time for quicker access.
opCallWav	Register	Calls the WAV file whose index is in the numeric register 'Register'
opLoadWav	Register	Loads the WAV file whose index is in the register 'Register', but does not call it. This can be used to prepare a WAV file ahead of time for quicker access.
opLoad	Register, Offset	Loads the property whose offset is 'Offset' into the register specified by 'Register'
opLoadAddress	Register, Offset	Loads the address of the item at offset 'Offset' into the specified numeric register.
opLoadImmediate	Register, Value	Loads the specified value into the specified register

opStore	Register, Offset	Stores the value in the specified register at the offset 'Offset' in the properties.
opAdd	Register1, Register2	Adds the value in numeric Register1 to the value in numeric Register2 and places the result in Register2
opSub	Register1, Register2	Subtracts the value in numeric Register1 from the value in numeric Register2 and places the result in Register2
opMult	Register1, Register2	Multiplies the value in numeric Register1 by the value in numeric Register2 and places the result in Register2
opDiv	Register1, Register2	Divides the value in numeric Register1 by the value in numeric Register2 and places the result in Register2
opMod	Register1, Register2	Takes the modulus of the value in numeric Register1 by the value in numeric Register2 and places the result in Register2
opAnd	Register1, Register2	Takes the boolean AND operation of the values in boolean Register1 and boolean Register2 and places the result in Register2
opOr	Register1, Register2	Takes the boolean OR operation of the values in boolean Register1 and boolean Register2 and places the result in Register2
opNot	Register1	Takes the boolean NOT operation of the value in boolean Register1 and places the result in Register1
opCat	Register1, Register2	Concatenates the strings in string registers Register1 and Register2 and leaves the result in Register2.
opNeg	Register1	Negates the value in numeric register Register1 and leaves the result in Register1
opCmpEQ	Register1, Register2, Register3	Sets boolean register Register1 to TRUE if the value in Register2 equals the value in Register3.
opCmpNE		Sets boolean register Register1 to TRUE if the value in Register2 is not equal to the value in Register3.
opCmpGT		Sets boolean register Register1 to TRUE if the value in Register2 is greater than the value in Register3.
opCmpLT		Sets boolean register Register1 to TRUE if the value in Register2 is less than the value in Register3.
opCmpGE		Sets boolean register Register1 to TRUE if the value in Register2 is greater than or equal to the value in Register3.
opCmpLE		Sets boolean register Register1 to TRUE if the value in Register2 is less than or equal to the value in Register3.
opBranchTrue	Register1, Offset	Branches to the specified offset if boolean register Register1 is TRUE
opBranchFalse	Register1, Offset	Branches to the specified offset if boolean register Register1 is FALSE
opWaitForEvent		Causes the system to pause until input of any kind is available to process. This input might include messages from the operating system, user input, mouse events, and so forth.

opWaitForUser

Causes the system to pause until the user types a keystroke or clicks the mouse. Useful for "Press any key to continue" messages.

APPENDIX B

## VRED DATABASE DEFINITION

## TABLE ClipExit

The ClipExit table is used to define the exits from a given clip to another clip or flow. There is one record for each transition between videos or for each flow which is called when a video frame is encountered. ClipExit specifies:

ID : an ID for this clipexit

FromClipID : the ID of the from clip

FromClipFrameNumber : the frame to trigger on which to trigger the clipexit

ToClipID : the destination clipID or nil if clip exit exists just to call a flow

ToClipFrameNumber : the target frame number in the destination clip

ToClipCalltype : either jump to target video or "call" target video

TransitionCode : a code to identify the type of transition to play between clips

FlowToRun : the ID of the flow to run for this clipexit

FlowCalltype : either jump to flow or "call" flow

ExitDirection : a symbolic which matches generic user input signals (eg left or right)

RoundaboutID: the ID for the roundabout if this clipexit is part of a roundabout (otherwise nil)

## TABLE ClipGroup

The ClipGroup table is used to define the paths in a map. There is one record for each path. The table SpatialClips records point to records in this table in a many to one relation. Each record specifies

ID : an ID for the the group of clips

ActiveClipID : the currently active clip which is drawn on the map

## TABLE Custom

The Custom table is used to define all the instances of Custom visuals in the production. It is joined with the visual table to determine each instance's geometry. Each record specifies:

ID : the ID of the custom visual (joined with visual

customIDType: the ID of the CustomType

## TABLE CustomType

The CustomType table is used to define all the custom visual types in the production. Each record specifies:

ID : The ID of the custom type

VisualType : a visual type numeric

TypeName : The name of the custom type

FileName : a handle to the file system for the file which implements the custom visual type.

## TABLE Flow

The Flow table is used to identify each of the flows in a production. Each record specifies:

ID : the ID for the Flow

Name : a text string name for this flow

StartFlowIconID : the ID for the starting token (FlowIcon) for this flow



## TABLE FlowIcon

The FFlowIcon table is used to store the parse tree for each flow in the production. Each record specifies:

ID : the ID for this FFlowIcon

ContainingFlowID : the ID for the Flow to which this flow icon belongs

IconType : the token type for this flowIcon in the flows parse tree

## TABLE FlowIconBooleanData

The FlowIconBooleanData table is used to define boolean data associated with a FlowIcon. Each record specifies:

FlowIconID : the ID for the FlowIcon to which the data element belongs

Ordinal : the sort order of all the FlowIconBooleanData records for this FlowIconID

BooleanDatum : the boolean datum value

## TABLE FlowIconConnection

The FlowIconConnection table is used to define the parse tree connectedness of the records in the FlowIcon table. Each record specifies

ParentIconID : the parent of the child node

Ordinal : the sort order of all the FlowIconConnection records

ChildIconID : the child of the parent node

## TABLE FlowIconIDData

The FlowIconIDData table is used to define ID data associated with a FlowIcon. Each record specifies:

FlowIconID : the ID for the FlowIcon to which the data element belongs

Ordinal : the sort order of all the FlowIconIDData records for this FlowIconID

IDDatum : the ID datum value

## TABLE FlowIconMemoData

The FlowIconMemoData table is used to define memo data (variable length string data) associated with a FlowIcon. Each record specifies:

FlowIconID : the ID for the FlowIcon to which the data element belongs

Ordinal : the sort order of all the FlowIconMemoData records for this FlowIconID

MemoDatum : the memo datum

## TABLE FlowIconNumericData

The FlowIconNumericData table is used to define numeric data (32-bit integers) associated with a FlowIcon. Each record specifies:

FlowIconID : the ID for the FlowIcon to which the data element belongs

Ordinal : the sort order of all the FlowIconNumericData records for this FlowIconID

NumericDatum : the numeric datum

## TABLE FlowIconStringData

The FlowIconStringData table is used to define numeric data (32-bit integers) associated with a FlowIcon. Each record specifies:

FlowIconID : the ID for the FlowIcon to which the data element belongs

Ordinal : the sort order of all the FlowIconStringData records for this FlowIconID

StringDatum : the string datum

## TABLE FramePosition

The FramePosition table is used to define the spatial coordinate data for each frame of a video clip. Because the camera is sometimes stationary, each record defines the frame at which the camera arrived at the point and the frame number at which the camera departed from the point. This table is also used to derive spline points from drawing clips on the map. One record exists for each video clip frame or spline point in a One-to-many relation from videoclip to FramePosition. Each record specifies:

ClipID : the ID of the video clip to which this frameposition record belongs

FrameIn : the first frame number for which this record defines a position

FrameOut : last frame number for which this record defines a position

X : the geographical point along the X-axis in millimeters

Y : the geographical point along the Y-axis in millimeters

Z : the geographical point along the Z-axis in millimeters

Bearing : the compass bearing of the camera

Pitch : the attitude of the camera

Roll : the roll of the camera

IsStraight : for splines, this value is a boolean indicating if the point is rounded or straight

RoundingFactor : a factor of the computation for rounding

## TABLE HotSpotByClip

The HotSpotByClip table is used to define the video hotspots. One record exist for each hotspot function on each video clip in the production. Each record specifies:

ID : the ID for this hotspot

Name : a user defined name for this hotspot

FlowID : the flow to run when this hotspot is triggered

Enabled : a control boolean for this hotspot

TriggerType : the user defined triggertype for this hotspot

FlowAccessType : the access type for the flow, either jump or call

clipID : the ID for the video clip which contains this hotspot

ClipExit : a clipexit to run when this hotspot is triggered

CursorID : the ID for the cursor to display when the mouse pointer is moved over this hotspot

## TABLE HotSpotByClipLocns

The HotSpotByClipLocns table is used to define the key frame locations of hotspots on each frame of video. One record exists for each hotspot on each frame. Each record specifies:

HotSpotID : the ID for the Hotspot in the Hotspotbyclip table

ClipID : the ID for the video clip which contains this hotspot

FrameNumber : the frame number to which this record applies

IsFirstFrame : an indicator if this is the first key frame

CenterPixelsX : the X coordinate of the center of the rectangle

CenterPixelsY : the Y coordinate of the center of the rectangle

WidthPixels : the width of the hotspot area

HeightPixels : the height of the hotspot area

## TABLE KeyboardInput

The KeyboardInput table is used to define the flows which are executed when the user presses specific keys or key sequences on the keyboard. Each record specifies

ID : the ID for this record

NonGraphicKey : a boolean indicating if this is a "printable" ascii character

FlowID : the flow to run when this key is pressed

FlowAccessType : the flow access type (jump or call)

Enabled : a boolean if this key response is enabled

VisualID : the ID for the visual which owns this key definition

GraphicString : a string of printable characters which maps to a sequence of key presses  
 ClipExit : the clipexit to activate when this KeyboardInput is matched

#### 5 TABLE Library

The Library table is used to define the file locations of all the assets used in the production. To construct the full record for each asset it is necessary to join one-to-one with each of the asset tables (videoclips where viewport is nil, bitmaps, or mview catalogs)

ID : the ID for this library record

10 string1 : a descriptive name for this asset

string2 : descriptive information about this asset

FileName : the handle to the file system location of the asset

MediaType : what asset type this record represents

MediaID : the ID of the record in each of the media tables

15 ClipListIdx : the index for sorting in the userinterface

ThumbnailFilename : the handle to the file system file which contains a an image to represent this asset

ResourceIndex : an index into the mview catalog for Mviews and albums

20

#### TABLE Map

The Map table is used to define the maps in the production. Each map represents a unique coordinate space. Each record in the map table specifies:

ID : the ID for the map record

25 Name : a user defined name for this map

OriginX : the X coordinate of the origin

OriginY : the Y coordinate of the origin

ViewportID : the default viewport for clips in this map

30

#### TABLE MIDI

The MIDI table is used to store information about MIDI assets. This table is joined with the library table to determine file locations. Each record specifies:

ID : the ID for this asset

35 LibraryID : the ID into the Library table for this asset

TitleIndex : an index to the MIDI catalog for this MIDI

Duration : the playtime length in seconds for this MIDI

Slot : a slot assignment for this MIDI

Looping : a boolean which defines if this MIDI file should be looped

40

#### TABLE Object

The Object table is used to define the objects in a production. Each record specifies:

ID : the ID for this object

45 Name : a user defined text name for this object

SuperClassObjectID : the parent class ID for the object

IsVirtual : a boolean which indicates whether this is an abstract class

NumberBitmapSets : the number of bitmaps in the Mview which represents this object

MviewFileName : the file name of the Mview catalog associated with this object

50 MviewIndex : the index in the Mview catalog associated with this object

#### TABLE OnScreenReceptacle

The OnScreenReceptacle table is used to define the receptacles for objects in the video. One record exist for each unique geographical point in space which can hold an object. Each record specifies:

55

ID : the ID for this receptacle

ObjectID : the ID for the initial object in the receptacle  
 Name : the user defined name for the receptacle  
 MapID : the ID for the map which contains this receptacle  
 Alignment : the alignment to be used when painting objects into this receptacle  
 MapX : the X coordinate in the map for this receptacle  
 MapY : the Y coordinate in the map for this receptacle  
 MapZ : the Z coordinate in the map for this receptacle

#### 10 TABLE OnScreenReceptacleLocns

The OnScreenReceptacleLocns table is used to define the key frame locations of receptacles on each frame of video. One record exists for each key frame definition of a receptacle. Each record specifies:

15 ReceptacleID : the ID for the receptacle which this point defines  
 ClipID : the ID for the video clip which contains this receptacle  
 FrameNumber : the frame number in the video clip which contains this receptacle key definition  
 IsFirstFrame : a boolean which indicates if this is the first key frame  
 PixelsX : the X pixel coordinate into the video for this receptacle  
 PixelsY : the Y pixel coordinate into the video for this receptacle  
 20 DrawScale : an arbitrary scale factor for applying to the rendering of objects in the receptacle  
 Bearing : the orientation face of the receptacle (0-259)

#### 25 TABLE Picture

The Picture table is used to define the picture frames for display of bitmaps. Pictures are a type of visual. Properties common to all visuals are stored in the Visual table and may be obtained by joining the Picture and Visual table.

ID : the ID for this picture frame  
 VisualID : the ID for the visual record which defines the geometry for this frame  
 30 Filename : the filename of the bitmap which contains this initial bitmap to be displayed  
 MviewFilename : the filename of the album catalog which contains the initial bitmap  
 AlbumIndex : the index in the album which contains the initial bitmap  
 PictureIndex : the picture index for the initial bitmap  
 LibraryID : the library record ID which contains file information for the initial bitmap  
 35

#### TABLE PropertyBoolean

The PropertyBoolean table is used to define the boolean properties of objects. Each record specifies:

ID : the ID for this property  
 40 Name : a user defined text name for this property  
 IsPerObject : a boolean which indicates if the property is per object or global  
 DefaultInitialValue : the default initial value

#### 45 TABLE PropertyMemo

The PropertyMemo table is used to define the memo properties of objects. Each record specifies:

ID : the ID for this property  
 Name : a user defined text name for this property  
 IsPerObject : a boolean which indicates if the property is per object or global  
 50 DefaultInitialValue : the default initial value

#### TABLE PropertyNumeric

The PropertyNumeric table is used to define the numeric properties of objects. Each record specifies:

55 ID : the ID for this property  
 Name : a user defined text name for this property

IsPerObject : a boolean which indicates if the property is per object or global  
 DefaultInitialValue : the default initial value

5 TABLE PropertyOfObjectBoolean

The PropertyOfObjectBoolean table is used to define the initial instance values for each object's properties. Each record specifies:

BooleanPropertyID : the ID of the property in the respective property table

ObjectID : the ID of the object

10 InitialValue : the default initial value for the object instance property

TABLE PropertyOfObjectMemo

The PropertyOfObjectMemo table is used to define the initial instance values for each object's properties. Each record specifies:

15 MemoPropertyID : the ID of the property in the respective property table

ObjectID : the ID of the object

InitialValue : the default initial value for the object instance property

20

TABLE PropertyOfObjectNumeric

The PropertyOfObjectNumeric table is used to define the initial instance values for each object's properties. Each record specifies:

NumericPropertyID : the ID of the property in the respective property table

25 ObjectID : the ID of the object

InitialValue : the default initial value for the object instance property

TABLE PropertyOfObjectString

30 The PropertyOfObjectString table is used to define the initial instance values for each object's properties. Each record specifies:

StringPropertyID : the ID of the property in the respective property table

ObjectID : the ID of the object

InitialValue : the default initial value for the object instance property

35

TABLE PropertyString

The PropertyString table is used to define the string properties of objects. Each record specifies:

ID : the ID for this property

40 Name : a user defined text name for this property

IsPerObject : a boolean which indicates if the property is per object or global

DefaultInitialValue : the default initial value

45 TABLE Roundabout

The Roundabout table is used to define the roundabouts (places you can turn around on a path) on a map. There is one record for for each clip in the clip group which contains the roundabout (eg four records if there is a forward, left facing, right facing and backwards clip). The actual turn is described by a collection of up to eight clipexit records which reference a specific roundabout. Each record specifies:

50

ID : The ID for the roundabout. All records for the same roundabout have the same ID.

MapID : The ID for the map which contains this round about

ClipID : The ID for the video clip which contains this round about

Frame : The frame on which this round takes place

55

## TABLE SpatialClip

The SpatialClip table is used to define properties of video clips which are specific to video which is used on a map (spatial video). Each record is left joined one-to-one with the VideoClip table. Each record specifies:

- ID : the ID for the VideoClip/Spatial Clip
- MapID : the ID for the map which contains this clip
- PositionFileID : the ID for the file which contains the CPA data
- GroupID : the ID for the group to which this clip belongs or nil if it is a single clip

## TABLE System

The System table is used to define global properties of the production which are used by the Edit system software. Each record specifies:

- GeneralPurposeLabel : a unique string label to identify the record
- GeneralPurposeString : a string value
- GeneralPurposeID : a numeric value

The contents of the system table should be:

GeneralPurposeLabel	GeneralPurposeString	GeneralPurposeID
ALLOCID		6176
DebugMode		0
STANDALONE		0
StartWindow		40

- AllocID - is a counter which indicates the last used ID in the production
- DebugMode - a boolean which is used by the binder to determine if debug capabilities should be turned on in the production
- StandAlone - a boolean used by the binder to determine if all software dependencies should be resolved or referenced
- StartWindow - an ID which indicates the startup visual

## TABLE SystemWindow

Table SystemWindow is used to store the edit system software window positions from sessions to sessions.

- WindowName : the string name of the window
- left : the left coordinate of the window in twips
- top : the top coordinate of the window in twips
- width : the width of the window in twips
- height : the height of the window in twips
- Visible : a boolean which indicates if the window is initially hidden or visible

## TABLE VideoClip

The VideoClip table is used to define all video in the production. Each record represents a video asset when left joined with the Library table. Each record represents a VideoClip on a map when left joined with the SpatialClip table

- ID : the ID for this Video
- Name : a user defined name which identifies this video
- FrameCount : the number of frames in a video asset
- MarkIn : the markin for a SpatialClip reference to the asset
- MarkOut : the markout for SpatialClip reference to the asset
- MidiID : the Library ID for the MIDI asset if this clip is accompanied by MIDI
- ViewPortID : the ID of the viewport visual into which this clip is played

LibraryID : the LibraryID for the file if this is a video asset

FrameRate : the frame rate of the video asset

Width : the X resolution of the video asset

Height : the Y resolution of the video asset

5 Anamorphic : a boolean which indicates if the video asset was shot with an anamorphic lense

SourceLeft : the source left coordinate of the rectangle used for projection into the viewport

SourceTop : the source top coordinate of the rectangle used for projection into the viewport

SourceRight : the source right coordinate of the rectangle used for projection into the viewport

SourceBottom : the source bottom coordinate of the rectangle used for projection into the viewport

10 PingPong : a boolean which indicates the video clip should be played forwards and backwards in a loop

ExitAction : the action of the runtime player when the last frame of the clip is played (loop or exit)

ParentVideoClipID : for single frame clips used in roundabouts, the ID of the parent clip from which this clips was extracted.

15

#### TABLE ViewPort

Table ViewPort is used to define the viewports in a production. There is one record for each viewport which is left joined with the visual table to determine the all the properties for the viewport visual.

20 Each record specifies:

ID : The ID for the viewport

LoadVideoID : the initial videoclip which is to be loaded and played when this viewport is activated

BackGroundColor : the background color to paint the viewport before drawing video

25 LeftTurnFlowID : the flow ID of the flow to execute when the user signals a left turn at run time

RightTurnFlowID : the flow ID of the flow to execute when the user signals a right turn at run time

CancelTurnFlowID : the flow ID of the flow to execute when the user signals to cancel a turn

#### TABLE Visual

30 The Visual table is used to define the common properties for each of the visual types (Viewport, Window, Picture, TextLabel etc.)

ID : the ID for the visual

ContainerID : the parent container for this visual

Name : a user defined text name for the visual

35 VisualType : an enumeration type for the visual type (viewport, window, picture, textlabel, userdefined)

Left : the left coordinate geometry for the visual

Top : the top coordinate geometry for the visual

Width : the width geometry for the visual

40 Height : the height geometry for the visual

MouseClickedFlowID : the ID of the flow to execute when the user clicks on this visual

MouseDownFlowID : the ID of the flow to execute when the user presses the mouse button down

MouseMoveFlowID : the ID of the flow to execute when the user moves the mouse over this visual

45 MouseUpFlowID : the ID of the flow to execute when the user releases the mouse button

Zorder : the zorder of this visual relative to its siblings in the container tree

Visible : initial boolean value for visibility

WindowID : the ID of the topmost parent window for this visual

ZorderCount : the highest used zorder of the child visuals in this visual

50

#### TABLE WallSegment

55 The WallSegment table is used to define the decorative walls drawn on a map. There is one record for each wall segment. Each record specifies:

MapID : The ID for the map which contains this wall

StartX : The geometry for the segment  
StartY : The geometry for the segment  
EndX : The geometry for the segment  
EndY : The geometry for the segment

5

#### TABLE Windows

The Windows table is used to define the Windows in a production. Windows are the type of all topmost visual containers in the visual hierarchy. There is one record for each window which is left joined with the visual table to determine the all the properties for the window visual. Each record specifies:

10

ID : the ID for the window visual

VisualID : the visual ID to join this record

BackgroundColor : the background paint color for initial display of the window

15

LoadFlowID : the ID of the flow to run when this visual is loaded

ZorderCount : the zorder of this window relative to other windows.



APPENDIX C

## MPEG STREAMER INTERFACE DEFINITION

```

5  typedef long SVPRC;          // Return code value
   typedef void *SVPSuperHandle; // Blind handle used by streamer

   // This structure defines an MPEG transition
   typedef struct tagMPEGTRANS
10  {
       long m_lLength;          // Length of data in m_pData
       unsigned char *m_pData; // Actual MPEG data for transition (all B frames),
                               // 'm_lLength' bytes long
       unsigned long m_ulFlags; // Flags, see MPTF_ defines below
15  short m_sNumFrames;         // Number of B-frames in this transition
       SVPRect *m_pPans;        // Amount by which to pan; m_sNumFrames elements or NULL
                               // if the MPTF_PANS bit is not on
   } MPEGTRANS;

20  #define MPTF_PANS 0x01 // Transition has panning information

   //
   // This structure is used for seeking and to indicate the current frame in any of the
   // frame sync functions. Note that when you call MpegSeekSMPTE or MpegSeekFrame, only the
25  // SMPTE or FrameNum portions are looked at for input and the other portion is appropriately
   // updated on return
   //
   typedef struct tagSYNCINFO {
30  long      m_lFrameNum;      // INOUT - Current frame number - 0 is the first frame
   BYTE      m_nSMPTEHour;     // INOUT - Hour component of SMPTE time code
   BYTE      m_nSMPTEMin;      // INOUT - Minute component of SMPTE time code
   BYTE      m_nSMPTESec;      // INOUT - Second component of SMPTE time code
   BYTE      m_nSMPTEFrame;    // INOUT - Frame count component of SMPTE time code
35  BYTE      m_nFrameType;     // INOUT - Type of the current FRAME (I, P, B, D)
   } SYNCINFO;

   #define MAX_PICS_PER_BUFFER 20 // Max number of pictures per MPPBuffer

   typedef struct MPPPicInfoTag
40  {
       long m_lPicNum;          // Picture number (0-origin)
       long m_lOffset;          // Offset of this picture in the buffer
   } MPPPicInfo;

45  // Codes for the m_nInUse field of MPPBuffer
   #define MPPBuffer_FREE 0 // Buffer is available
   #define MPPBuffer_PLAYER 1 // Buffer is in use by player
   #define MPPBuffer_UPDATE 2 // Buffer is being updated, do not use

50  typedef struct MPPBufferTag
   {
       struct MPPBufferTag *m_pNext; // Next buffer in chain (used by streamer only)
       void *m_pvData;               // MPEG data
       long m_lFileOffset;           // File offset of MPEG data
55  long m_lLength;                  // Length of MPEG data
   }

```

```

    long m_lMax;      // Maximum length allocated
    long m_lTag;      // Tag of MPPBuffer
    long m_lNumFrames; // Number of frames in MPEG data
    int m_nInUse;      // See MPPBuffer_codes above
5    MPPPicInfo m_PicInfo[MAX_PICS_PER_BUFFER];
        // List of picture number/offset combinations,
        // sorted by offsets.
} MPPBuffer;

10 // MPSOpen initializes the streamer on the given file. The file will
    // usually be the clip hunk generated by the binder.
    SVPRC MPSOpen(SVPSuperHandle &sup, const char *strFileName);

    // MPSClose closes a previously-opened file
15 SVPRC MPSClose(SVPSuperHandle sup);

    // MPSSeek seeks the currently-open stream to the given file number
    SVPRC MPSSeek(SVPSuperHandle sup, long lFrameNum);

20 // MPSCacheFrame copies the specified frame number into the frame
    // cache without displaying it. Typically the cache ID is the first
    // frame of the actual TO video clip, while the lFrameNum parameter
    // is the frame number of the cached copy of the TO frame that the
    // binder copies into the clip hunk.
25 SVPRC MPSCacheFrame(SVPSuperHandle sup, long lFrameNum, long lCacheID);

    // MPSDeCacheFrame removes a previously-cached frame from the cache.
    SVPRC MPSDeCacheFrame(SVPSuperHandle sup, long lCacheID);

30 // MPSDeCacheAll removes all previously-cached frames from the cache
    SVPRC MPSDeCacheAll(SVPSuperHandle sup);

    // MPSPlayTransition causes a transition to play using the MPEG frame
    // that is currently in the future buffer as the FROM frame and the
35 // specified cached frame as the TO frame. The transition to use is
    // specified by the pMpegTrans structure passed in.
    SVPRC MPSPlayTransition(SVPSuperHandle sup,
        MPEGTRANS *pMpegTrans,
        long lCacheID,
40        long lSeekFrame == -1);

    // MPSWork must be called periodically to give the streamer a chance to
    // read data and to give a software player, if one is being used, a
    // chance to do its work.
45 SVPRC MPSWork(SVPSuperHandle sup, SYNCINFO *pSyncInfo);

    // MPSChopStream terminates the currently-playing MPEG stream after
    // the specified frame number. Used for eliminating information that
    // has already been read but has not yet been sent to the MPEG
50 // player in preparation for exiting the current clip.
    SVPRC MPSChopStream(SVPSuperHandle sup, long lFrameNum);

    // MPSHoldVideo is used to prevent the streamer from reading any more
    // video from the clip hunk. Used before an MPSChopStream call to
55 // make sure the streamer doesn't read any more video before the
    // transition takes place.

```

SVPRC MPSholdVideo(SVPSuperHandle sup, int hold);

TABLE 1


---

```

-----Logic Hunk-----
5 00000000:Source Line: [---- System ----]
   00000018:LoadWin          INDX:00000001
   00000020:WaitForEvent
   00000024:GOTO             ADDR:00000020
   0000002c:EXIT

```

---

10

TABLE 2


---

```

15 00000030:Source Line: [22 tables facing south ----]
   00000053:VIDEO_LOGIC_START  FRAM:      0,RETN:0000082f
   0000005f:FRAM:              0,FRAME_POSITION      ,X -1740 Y -1245 Z      0 B      0
P      0 R      0
   00000073:FRAM:              20,HOTSPOT_ACTIVATE    ,IDX :      27,SIDX:      0
20 00000087:FRAM:             162,FRAME_POSITION      ,X -1721 Y -1242 Z      0 B      0
P      0 R      0
   ...
   0000028f:FRAM:             318,FRAME_POSITION      ,X -1223 Y -1161 Z      0 B      0
P      0 R      0
25 000002a3:FRAM:             318,HOTSPOT_DEACTIVATE ,IDX :      27,SIDX:      0
   000002b7:FRAM:             324,FRAME_POSITION      ,X -1204 Y -1158 Z      0 B      0
P      0 R      0
   ...
30 00000807:FRAM:             729,FRAME_POSITION      ,X   90 Y  -945 Z      0 B      0
P      0 R      0
   0000081b:FRAM:             730,CLIP_END
   0000082f:RETURN

```

---

TABLE 3

```

00000833:Source Line: [24 EastAcrossFront ----]
00000852:VIDEO_LOGIC_START   FRAM:      741,RETN:00000fa2
5  0000085e:FRAM:      741,FRAME_POSITION   ,X   105 Y  -945 Z      0 B      0
    P      0 R      0
00000872:FRAM:      741,CACHE_FRAME           ,DEST:    1269,CACH:    732
00000886:FRAM:      741,CACHE_FRAME           ,DEST:    2194,CACH:    733
0000089a:FRAM:      741,CACHE_FRAME           ,DEST:   14586,CACH:    734
10 000008ae:FRAM:      741,CACHE_FRAME           ,DEST:   14399,CACH:    735
    000008c2:FRAM:      741,CACHE_FRAME           ,DEST:   24849,CACH:    736
    000008d6:FRAM:      741,CACHE_FRAME           ,DEST:   14376,CACH:    737
    000008ea:FRAM:      741,CACHE_FRAME           ,DEST:   24854,CACH:    738
    000008fe:FRAM:      741,CACHE_FRAME           ,DEST:   14554,CACH:    739
15 00000912:FRAM:      741,CACHE_FRAME           ,DEST:   24859,CACH:    740
    00000926:FRAM:      741,RECEPTACLE_ACTIVATE ,      7,      1
    0000093a:FRAM:      744,HOTSPOT_ACTIVATE   ,IDX :      26,SIDX:      0
    0000094e:FRAM:      746,HOTSPOT_ACTIVATE   ,IDX :     136,SIDX:      0
    00000962:FRAM:      747,FRAME_POSITION   ,X    75 Y  -945 Z      0 B      0
20  P      0 R      0
    00000976:FRAM:      747,RUNTIME_TRANSITION ,EXID:002e8/03,TO   :
    14586,TRAN:      1
    0000098a:FRAM:      747,CLIP_EXIT
    ,EXID:002e8/03,ADDR:000128f6,JUMP_ACCESS_TYPE
25 0000099e:FRAM:      753,FRAME_POSITION   ,X    44 Y  -945 Z      0 B      0
    P      0 R      0
    ...
    00000a3e:FRAM:      801,FRAME_POSITION   ,X   -204 Y  -945 Z      0 B      0
    P      0 R      0
30 00000a52:FRAM:      806,RUNTIME_TRANSITION ,EXID:005b7/03,TO   :
    14554,TRAN:      2
    00000a66:FRAM:      806,CLIP_EXIT
    ,EXID:005b7/03,ADDR:000128f6,JUMP_ACCESS_TYPE
    00000a7a:FRAM:      806,RUNTIME_TRANSITION ,EXID:005b8/04,TO   :
35 24859,TRAN:      0
    00000a8e:FRAM:      806,CLIP_EXIT
    ,EXID:005b8/04,ADDR:000274b9,CALL_ACCESS_TYPE
    00000aa2:FRAM:      806,CLIP_EXIT
    ,EXID:005b8/04,ADDR:0001eccd,JUMP_ACCESS_TYPE
40 00000ab6:FRAM:      807,FRAME_POSITION   ,X   -235 Y  -945 Z      0 B      0
    P      0 R      0
    ...
    00000af2:FRAM:      825,FRAME_POSITION   ,X   -328 Y  -945 Z      0 B      0
    P      0 R      0
45 00000b06:FRAM:      827,RECEPTACLE_DEACTIVAT,IDX :      7,SIDX:      1
    00000b1a:FRAM:      831,FRAME_POSITION   ,X   -359 Y  -945 Z      0 B      0
    P      0 R      0
    00000b2e:FRAM:      835,HOTSPOT_DEACTIVATE ,IDX :     136,SIDX:      0
    00000b42:FRAM:      837,FRAME_POSITION   ,X   -390 Y  -945 Z      0 B      0
50  P      0 R      0
    00000b56:FRAM:      843,FRAME_POSITION   ,X   -421 Y  -945 Z      0 B      0
    P      0 R      0
    ...
    00000bce:FRAM:      879,FRAME_POSITION   ,X   -607 Y  -945 Z      0 B      0

```

```

P      0 R      0
00000be2:FRAM:      884,RUNTIME_TRANSITION ,EXID:00034/04,TO :
1269,TRAN:      0
5 00000bf6:FRAM:      884,CLIP_EXIT
,EXID:00034/04,ADDR:00000fa6,JUMP_ACCESS_TYPE
00000c0a:FRAM:      885,FRAME_POSITION ,X -638 Y -945 Z      0 B      0
P      0 R      0
00000c1e:FRAM:      886,RUNTIME_TRANSITION ,EXID:00172/03,TO :
2194,TRAN:      1
10 00000c32:FRAM:      886,CLIP_EXIT
,EXID:00172/03,ADDR:0000231f,JUMP_ACCESS_TYPE
00000c46:FRAM:      891,FRAME_POSITION ,X -669 Y -945 Z      0 B      0
P      0 R      0
...
15 00000ce6:FRAM:      939,FRAME_POSITION ,X -917 Y -945 Z      0 B      0
P      0 R      0
00000cfa:FRAM:      940,HOTSPOT_DEACTIVATE ,IDX :      26,SIDX:      0
00000d0e:FRAM:      945,FRAME_POSITION ,X -948 Y -945 Z      0 B      0
P      0 R      0
20 ...
00000e76:FRAM:      1053,FRAME_POSITION ,X -1505 Y -945 Z      0 B      0
P      0 R      0
00000e8a:FRAM:      1053,RUNTIME_TRANSITION ,EXID:005a5/03,TO :
14399,TRAN:      2
25 00000e9e:FRAM:      1053,CLIP_EXIT
,EXID:005a5/03,ADDR:000128f6,JUMP_ACCESS_TYPE
00000eb2:FRAM:      1053,RUNTIME_TRANSITION ,EXID:005a6/04,TO :
24849,TRAN:      0
00000ec6:FRAM:      1053,CLIP_EXIT
30 ,EXID:005a6/04,ADDR:0001eb37,JUMP_ACCESS_TYPE
00000eda:FRAM:      1059,FRAME_POSITION ,X -1536 Y -945 Z      0 B      0
P      0 R      0
00000eee:FRAM:      1065,FRAME_POSITION ,X -1567 Y -945 Z      0 B      0
P      0 R      0
35 00000f02:FRAM:      1071,FRAME_POSITION ,X -1598 Y -945 Z      0 B      0
P      0 R      0
00000f16:FRAM:      1077,FRAME_POSITION ,X -1629 Y -945 Z      0 B      0
P      0 R      0
00000f2a:FRAM:      1078,FRAME_POSITION ,X -1635 Y -945 Z      0 B      0
40 P      0 R      0
00000f3e:FRAM:      1079,RUNTIME_TRANSITION ,EXID:005af/03,TO :
14376,TRAN:      2
00000f52:FRAM:      1079,CLIP_EXIT
,EXID:005af/03,ADDR:000128f6,JUMP_ACCESS_TYPE
45 00000f66:FRAM:      1079,RUNTIME_TRANSITION ,EXID:005b0/04,TO :
24854,TRAN:      0
00000f7a:FRAM:      1079,CLIP_EXIT
,EXID:005b0/04,ADDR:0001ec02,JUMP_ACCESS_TYPE
00000f8e:FRAM:      1079,CLIP_END
50 00000fa2:RETURN

```

TABLE 4a


---

```

5  Flow ShowGriffey
   assign bbtext.contents=picGriffey
   visual show bbtext
   waitforuser
   visual hide bbtext
   End Flow
10

```

---

15

20

TABLE 4b


---

```

25 000226f4:Source Line: [1259 ShowGriffey]
    0002270c:FLOW_LOGIC_START
    00022710:Source Line: [bbtext.contents = picGriffey]
    00022734:LoadImmediate      TYPE:2      ,NREG:0      ,VAL :      0
    00022744:LoadImmediate      TYPE:2      ,NREG:1      ,VAL :    65539
    00022754:Load               TYPE:2      ,NREG:3      ,ADDR:000000c6
30 00022764:StoreVisual         TYPE:2      ,&VIS:00000001,&DSP:0
    &VAL:3
    00022778:ShowPicture         INDX:00000001
    00022780:WaitForUser
    00022784:HidePicture         INDX:00000001
35 0002278c:RETURN

```

---

TABLE 5

---

```

-----Hotspot SubHunk-----
VersionID 7
5 FirstHotspotRegionOffset 0x3890
  HotspotRegionBlockSize 0
  HotspotRegionCount 0
  HotspotRegionSize 36
10 FirstHotspotClipOffset 0x3890
  HotspotClipBlockSize 192556
  HotspotClipCount 328
  HotspotClipSize 28

  HotspotRegionCount 0
15 HotspotClipCount 328
    -----HotspotClip 0-----
    OccurrenceCount 1
    OccurrenceSize 12
20 FirstOccurrenceOffset 0x38ac
    FlowAddress 0x21b89
    FlowAccessType 1
    ExitID -1
    TriggerType 0
25 State 1
    CursorID -1
        -----HotspotClip 0 Occurrence 0-----
        ActivateFrame 4820
        DeactivateFrame 4914
30 FirstHotspotFrameOffset 0x38b8

the interesting part is here; it is a list of center,width,height on a
frame-by-frame basis for this hotspot.

35 0: CenterX=229 CenterY=126 HalfWidth=8 HalfHeight=25
   ... 92 not printed
   93: CenterX=324 CenterY=189 HalfWidth=26 HalfHeight=50
   -----HotspotClip 1-----
   OccurrenceCount 1
40 OccurrenceSize 12
   FirstOccurrenceOffset 0x3bc4
   FlowAddress 0x21b17
   FlowAccessType 1
   ExitID -1
45 TriggerType 0
   State 1
   CursorID -1
       -----HotspotClip 1 Occurrence 0-----
       ActivateFrame 4820
50 DeactivateFrame 4908
       FirstHotspotFrameOffset 0x3bd0
       0: CenterX=244 CenterY=134 HalfWidth=7 HalfHeight=32
       ... 86 not printed
       87: CenterX=352 CenterY=187 HalfWidth=14 HalfHeight=49

```



```
-----HotspotClip 2-----
OccurrenceCount 1
OccurrenceSize 12
FirstOccurrenceOffset 0x3eac
5 FlowAddress 0x21aa6
FlowAccessType 1
ExitID -1
TriggerType 0
State 1
10 CursorID -1
    -----HotspotClip 2 Occurrence 0-----
    ActivateFrame 4820
    DeactivateFrame 4883
    FirstHotspotFrameOffset 0x3eb8
15     0: CenterX=263 CenterY=144 HalfWidth=10 HalfHeight=37
    ... 61 not printed
    62: CenterX=353 CenterY=203 HalfWidth=12 HalfHeight=40
    -----HotspotClip 3-----
    OccurrenceCount 1
20 OccurrenceSize 12
    FirstOccurrenceOffset 0x40cc
    FlowAddress 0x21b89
    FlowAccessType 1
    ExitID -1
25 TriggerType 1
    State 1
    CursorID -1
    -----HotspotClip 3 Occurrence 0-----
    ActivateFrame 4100
30 DeactivateFrame 4229
    FirstHotspotFrameOffset 0x40d8
    0: CenterX=151 CenterY=112 HalfWidth=8 HalfHeight=4
    ... 127 not printed
    128: CenterX=29 CenterY=248 HalfWidth=35 HalfHeight=31
35
```

---

CLAIMS

We claim:

1. A method of developing an interactive multimedia production, comprising the steps of:
  - a. capturing the appearance of at least one physical or virtual environment with a motion picture camera, motion video camera or a virtual camera traveling along at least one natural path within the environment; and
  - b. using the captured appearance of the environment to create a navigable motion video representation of the environment.
2. A method according to claim 1, wherein the appearance of the environment is captured along a plurality of paths, at least two of which intersect.
3. A method according to claim 1, wherein the field of view captured at any given point along a path in the environment is greater than the field of view to be displayed at any one time.
4. A method according to claim 3, wherein the total field of view at a given point along the path is captured at once by means of a wide-angle lens or its virtual equivalent.
5. A method according to claim 3, wherein the horizontal field of view captured at any given point along a path in the environment is greater than the horizontal field of view to be displayed at any one time.
6. A method according to claim 3, wherein the total field of view captured at any point along a path is captured by causing the camera or virtual camera to travel the path more than one time, wherein every time the camera or virtual camera travels the path, the camera heading is altered so as to capture a different field of view, but the camera substantially retraces the same X, Y and Z coordinate positions of each point along the path.

7. A method according to claim 6, wherein the heading of the camera or virtual camera is changed each time the camera travels the path by rotating the camera or virtual camera substantially around a single axis of rotation.
8. A method according to claim 7, wherein said axis of rotation passes through or near the nodal point of the lens.
9. A method according to claim 7, wherein the axis of rotation is substantially perpendicular to the direction of travel along the path.
10. A method according to claim 1, wherein the camera lens path is at a height approximating the height of the head of a person navigating the environment.
11. A method according to claim 7, wherein the field of view captured by said camera or said virtual camera is captured by means of a wide-angle lens or its virtual equivalent.
12. A method according to claim 11, wherein the lens captures a horizontal field of view of at least about ninety degrees with each pass of the camera or virtual camera.
13. A method according to claim 12, wherein four camera headings, each offset from the adjacent heading by about ninety degrees, are selected so that the entire three hundred sixty degree field of view around the camera's axis of rotation from all of the points along a path in the environment is captured with four camera trips along the path.
14. A method according to claim 13, wherein:
  - (a) one of the four camera headings is along the path from the first end of the path to the second end of a path;
  - (b) a second camera heading is along the path from the second end of the path to the first end of the path; and

- (c) the remaining two camera headings are respectively ninety degrees to the left and ninety degrees to the right of the path from the first end of the path to the second end of a path.
- 15. A method according to claim 6, wherein the camera headings are selected so that the frames or virtual frames of visual information captured at a given point along the path with adjacent headings can be seamlessly joined side by side.
  - 16. A method according to claim 7, wherein the camera headings are selected so that the total visual information captured at a given point along the path comprises a 360 degree field of view around the camera's axis of rotation.
  - 17. A method according to claim 11, wherein the headings and lens characteristics of the camera traveling the path are selected so that the frames taken with adjacent headings from a single point on the path can be seamlessly joined side by side, creating a 360 degree field of view around the axis of rotation of the camera.
  - 18. A method according to claim 17, wherein the field of view captured by said camera or said virtual camera is captured by means of a lens which results in images that together form a substantially cylindrical projection image.
  - 19. A method according to claim 17, wherein the field of view captured by said camera or said virtual camera is captured by means of a lens which results in images that are capable of being connected together by post-capture processing to form substantially cylindrical projection.
  - 20. A method according to claim 1, wherein a plurality of environments are captured, which environments may be in non-adjacent coordinate spaces.
  - 21. A method according to claim 20, wherein separately captured environments are merged into one larger environment.

22. A method according to claim 21, wherein the coordinates of the separately captured environments are normalized in relation to each other when the environments are merged.
23. A method according to claim 20, wherein at least two of the plurality of environments are made jointly navigable by the creation of at least one transition between them.
24. A method according to claim 1, further comprising the steps of:
  - (a) capturing camera position data or virtual camera position data as the camera travels each natural path; and
  - (b) correlating the captured camera position data with the film, video, or virtual frames taken while the camera is in that position.
25. A method according to claim 24, wherein the camera position data captured comprises the X and Y coordinates of the camera's position.
26. A method according to claim 25, wherein the camera position data captured further comprises the Z coordinate of the camera's position.
27. A method according to claim 24, wherein the camera position data captured comprises the camera azimuth, pitch and roll.
28. A method according to claim 24, wherein at least one element of the camera position data is determined by means of an ultrasonic positioning system.
29. A method according to claim 28, wherein said positioning system comprises:
  - (a) an ultrasonic transmitter co-located with the camera; and
  - (b) a plurality of ultrasonic receivers at various locations within the environment.

30. A method according to claim 29, wherein at least one of the X, Y, and Z elements of the camera position data is calculated using a push-pull algorithm.
31. A method according to claim 24, wherein the positional data is calculated in real time.
32. A method according to claim 24, wherein the correlation step further comprises correlation of a particular frame with position data generated at the time the frame is shot.
33. A method according to claim 24, wherein the position data is generated manually by inspection of the film, video, or virtual frames.
34. A method according to claim 32, wherein the correlation step is accomplished by associating the same time code with the positional data and the frame.
35. A method according to claim 34, wherein the time code is an SMPTE time code.
36. A method according to claim 34, wherein the position data is time-stamped with the time code using the camera's time code generator or a time code generator synchronized with the camera's time code generator.
37. A method according to claim 1, further comprising the step of capturing position data for features in the environment by means of a portable transmitter which can be placed on or in the vicinity of the feature whose position is to be captured.
38. A method according to claim 37, wherein the portable transmitter is ultrasonic.
39. A method according to claim 34, wherein the time code associated position data is stored in a database.

40. A method of developing an interactive multimedia production, comprising the steps of:
  - a. using a motion camera to capture a field of view of at least one environment at a plurality of points along at least one natural path within the environment; and
  - b. creating a navigable motion video representation of the environment based upon the captured fields of view.
41. A method according to claim 40, wherein said motion camera is a motion picture camera.
42. A method according to claim 40, wherein said motion camera is a motion video camera.
43. A method according to claim 40, wherein said environment is a virtual environment and said motion camera is a virtual camera.
44. A method according to claim 40, wherein said field of view is a 360° field of view.
45. A method according to claim 40, wherein said field of view is a substantially cylindrical projection image.
46. A method according to claim 40, wherein said field of view is a substantially spherical projection image.
47. A method according to claim 40, wherein said field of view of the environment at each point is obtained by capturing a plurality of images with a field of view less than the total field of view captured.

48. A method according to claim 47, wherein each of said plurality of images has a horizontal field of view of about 90° and is directionally offset from the adjacent view by about 90°.
49. A method according to claim 47, wherein said plurality of images together form a 360° field of view.
50. A method according to claim 47, wherein said plurality of images are captured using an approximately cylindrical projection lens.
51. A method according to claim 47, wherein said plurality of images are captured using a wide-angle lens.
52. A method according to claim 47, wherein said plurality of images are captured in a manner to allow the images to be converted into planar projections by means of post-capture processing.
53. A method according to claim 47, wherein said plurality of images are captured in a manner to allow the images to be converted into cylindrical projections by means of post-capture processing.
54. A method according to claim 50, further comprising the step of removing any spherical distortion from each of said plurality of images.
55. A method according to claim 47, wherein said plurality of images can be stitched together to form a seamless 360° view.
56. A method according to claim 47, wherein there is no substantial overlap and no substantial underlap among said plurality of images



57. A method according to claim 47, wherein said plurality of images are obtained by moving the camera along the natural path a plurality of times with the camera facing in a different direction each time.
58. A method according to claim 47, wherein the nodal point of the camera is at approximately the same location when capturing each of said plurality of images from a particular point.
59. A method according to claim 47, wherein the camera's axis of rotation remains constant as it captures each of said plurality of images from a particular point.
60. A method according to claim 40, further comprising the steps of:
  - (a) recording position data for the camera as it moves within the environment; and
  - (b) associating said position data with the image captured at the corresponding position.
61. A method according to claim 60, wherein said position data comprises the X and Y coordinates of the camera within the environment.
62. A method according to claim 61, wherein said position data further comprises the Z coordinate of the camera within the environment.
63. A method according to claim 61, wherein said position data further comprises the azimuth, pitch, and roll of the camera.
64. A method according to claim 60, further comprising the step of recording position data for the significant features within the environment.
65. A method according to claim 60, wherein said position data is obtained using an ultrasonic trilateration technique.

66. A method according to claim 65, wherein said ultrasonic trilateration technique employs a push-pull algorithm to minimize the effect of errors and false signals.
67. A method according to claim 40, wherein said step of creating a navigable motion video representation of the environment comprises the steps of:
- (a) creating a map of the camera paths through the environment;
  - (b) editing the production using a map-based computer editing system; and
  - (c) binding the production.
68. A method according to claim 67, wherein said editing system allows a plurality of users of the system to edit the production simultaneously.
69. A method according to claim 67, wherein said editing system allows for the addition of hotspots to the captured views of the environment.
70. A method according to claim 69, wherein:
- (a) said editing system allows a user to define certain features within the environment to be associated with a hotspot; and
  - (b) said editing system is capable of automatically determining the proper position and size of the hotspot in the views of the environment based upon actual or virtual camera position data and a 3-D model of the environment.
71. A method according to claim 69, wherein:
- a. said editing system allows a user of the system to define the position and size of a hotspot in non-adjacent views along a path; and
  - b. said editing system is capable of interpolating the position and size of the hotspot in views along the path between the views for which the position and size of the hotspot has been defined by the user.

72. A method according to claim 67, wherein said editing system allows a user of the system to add receptacles to the captured views of the environment.
73. A method according to claim 72, wherein:
- (a) said editing system allows a user to define certain features within the environment to be associated with a receptacle; and
  - (b) said editing system is capable of automatically determining the proper position and scale factor of the receptacle in the views of the environment based upon actual or virtual camera position data and a 3-D model of the environment.
74. A method according to claim 72, wherein:
- a. said editing system allows a user of the system to define the position and scale factor of a receptacle in non-adjacent views along a path; and
  - b. said editing system is capable of interpolating the position and scale factor of the receptacle in views along the path between the views for which the position and size of the receptacle has been defined by the user.
75. A method according to claim 72, wherein said editing system allows for the addition of M-views to the captured views of the environment.
76. A method according to claim 75, wherein said editing system is capable of automatically determining the optimal number of faces and angles for the M-views based upon actual or virtual camera position data and a 3-D model of the environment.
77. A method according to claim 67, wherein said editing system allows for the association of dramatic video clips with certain locations on said map.
78. A method according to claim 67, wherein said editing system allows for the association of audio clips with certain locations on said map.

79. A method according to claim 67, wherein said editing system allows for the association of dramatic video clips with certain events which may occur during execution of the production at runtime.
80. A method according to claim 67, wherein said editing system allows for the association of audio clips with certain events which may occur during execution of the production at runtime.
81. A method according to claim 67, wherein representations of a plurality of environments are captured and said editing system allows for the connection of any captured representation of an environment with any other by use of transitions.
82. A method according to claim 81, wherein a connecting transition is associated with a specific feature in each environment such that the execution of the transition creates the appearance of a physical connection between the two environments.
83. A method according to claim 81, wherein said representations of the environments are not actually physically connected.
84. A method according to claim 81, wherein at least one of the representations of the environments is rendered.
85. A method according to claim 81, wherein at least one of the representations of the environments is captured from a physically existing environment.
86. A method of creating a navigable motion video production comprising the steps of:
  - (a) creating a map-based script including at least one natural path through the navigable environment;
  - (b) constructing or selecting at least one environment to be captured;

- (c) capturing the environment with a motion picture camera, motion video camera or a virtual camera;
  - (d) using a computer controlled map-based navigational motion video editing system to edit the production; and
  - (e) binding the production.
87. A method according to claim 86, wherein said editing system is capable of allowing a plurality of users of the system to edit the production simultaneously.
88. A method according to claim 86, wherein each path through the environment is set out on at least one two-dimensional map.
89. A method according to claim 88, wherein the paths are set out on at least one two-dimensional map display in a computer-based navigable motion video production authoring and editing system.
90. A method according to claim 89, wherein the authoring and editing system includes means for displaying a video frame captured from a given location in the environment in response to a user selecting a position on the map corresponding to the location in the environment.
91. A method according to claim 89, wherein the authoring and editing system includes means for displaying a given position on the map in response to a user selecting a video frame captured from a location in the environment corresponding to the position on the map.
92. A method according to claim 89, wherein the authoring and editing system includes means for laying out features the author wishes to be included in the navigable motion video environment in advance of the creation or capture of the features.

93. A method according to claim 92, wherein the features laid out in advance of their creation or capture are represented by symbols on the two dimensional map in proper relationship to each other and to the paths.
94. A method according to claim 93, wherein the features comprise hotspots, receptacles and objects.
95. A method according to claim 93, wherein the features further comprise entrances, path exits, scene exits, path intersections, music, and dramatic video.
96. A method according to claim 95, wherein said scene exits act as transitions between two different maps.
97. A method according to claim 93, wherein the laying out means prompts the author to select the type of feature being placed on the map and to describe its properties and interactions.
98. A method according to claim 97, wherein the description of properties is accomplished at least in part by selecting properties from a pre-created list.
99. A method according to claim 97, wherein the description of properties is accomplished at least in part by identifying the type of feature and textually describing its properties.
100. A method according to claim 97, wherein the authoring and editing system further comprises means for correlating at least one record in a database to each path and feature placed in the authoring and editing system.
101. A method according to claim 100, wherein the authoring and editing system further comprises means for creating program logic and associating it with features laid out in the authoring and editing system.

102. A method according to claim 101, wherein the authoring and editing system further comprises means for creating at least one record in a database for every program logic flow and for its association with features.
103. A method according to claim 102, wherein the authoring and editing system comprises means for playing the production during the authoring process prior to its completion, with symbols or textual descriptions substituting for the features and assets and with program logic functioning.
104. A method according to claim 103, wherein the authoring and editing system further comprises means for replacing the symbols and textual descriptions in the database with references to video assets, hotspot definitions and other actual features and assets as the features and assets are produced and entered into the computer-based system.
105. A method according to claim 89, wherein the authoring and editing system comprises means for identifying assets and features required to be captured or otherwise created for incorporation into the multimedia production.
106. A method according to claim 88, wherein each path is generated by camera position data.
107. A method according to claim 88, wherein each path is drawn on the map in the computer.
108. A method according to claim 88, wherein the map in the computer and at least one path represented thereon is created from a map and path earlier created outside the computer and then traced on a digitizing tablet or otherwise entered into the system's map function

109. A method according to claim 89, wherein the authoring and editing system comprises means for associating at least one video clip representing at least one camera heading of a path with the map clip, wherein said associating means includes means for selecting the map clip, means for selecting video clips for associating with the map clip, means for selecting the IN frame and the OUT frame for each clip associated with the map clip, and means for associating each IN frame and OUT frame with the first end and the second end of the map clip.
110. A method according to claim 109, wherein the video clips associated with the map clip form a group of clips.
111. A method according to claim 109, wherein the authoring and editing system further comprises transition creating means for creating transitions between frames in the same video clip.
112. A method according to claim 111, wherein said authoring and editing system allows a plurality of users of the system to edit transitions simultaneously.
113. A method according to claim 110, wherein the authoring and editing system further comprises transition creating means for creating transitions between frames in different video clips within the same group of clips.
114. A method according to claim 110, wherein the authoring and editing system further comprises transition creating means for creating transitions between a frame in a video clip associated with one map and a frame in a video clip associated with a second map.
115. A method according to claim 113, wherein said authoring and editing system allows a plurality of users of the system to edit transitions simultaneously.



116. A method according to claim 109, wherein the authoring and editing system further comprises transition generating means for generating transitions between a frame in a video clip associated with one map clip and a frame in a video clip associated with an intersecting map clip.
117. A method according to claim 109, wherein the authoring and editing system further comprises transition creating means for creating transitions between a frame in a navigable motion video clip and a dramatic video clip.
118. A method according to claim 109, wherein the authoring and editing system further comprises transition generating means and previewing means whereby the user may preview the created transition at edit time.
119. A method according to claim 109, wherein said associating means comprises means for creating database entries which relate the video clips and the IN frames and OUT frames to the map clip and to each other.
120. A method according to claim 109, wherein the authoring and editing system further comprises means for identifying map clip intersections.
121. A method according to claim 110, wherein the authoring and editing system further comprises means for creating roundabout transitions at any point along the map clip.
122. A method according to claim 120 wherein the authoring and editing system further comprises means for creating turn transitions between video clips in intersecting groups of clips at map clip intersections.
123. A method according to claim 121, wherein said roundabout transition creating means comprises:
  - (a) means for selecting FROM and TO clips from two different video clips;
  - (b) means for selecting the direction of the transition;

- (c) means for reviewing individual frames in the FROM clip side by side with individual frames in the TO clip to allow the selection of the closest matching frames for each transition; and
  - (d) means for selecting an individual frame in the TO clip as the TO frame and an individual frame in the FROM clip as the FROM frame.
124. A method according to claim 122, wherein said turn transition creating means comprises:
- (a) means for selecting FROM and TO clips from two different video clips, each of which is associated with one of two intersecting map clips;
  - (b) means for selecting the direction of the transition;
  - (c) means for reviewing individual frames in the FROM clip side by side with individual frames in the TO clip to allow the selection of the closest matching frames for each transition; and
  - (d) means for selecting an individual frame in the TO clip as the TO frame and an individual frame in the FROM clip as the FROM frame.
125. A method according to claim 123, wherein said FROM and TO clips are selected from clips in the same group of clips.
126. A method according to claim 124, wherein said FROM and TO clips are selected from clips in two different groups of clips.
127. A method according to claim 123, wherein the authoring and editing system further comprises means for using camera position data associated with each frame in the FROM and TO clip to provide the editor with the most likely matching frames for each requested roundabout transition.
128. A method according to claim 124, wherein the authoring and editing system further comprises means for using camera position data associated with each frame in the

FROM and TO clip to provide the editor with the most likely matching frames for each requested turn transition.

129. A method according to claim 123, wherein the authoring and editing system further comprises means for using the mark in and mark out points of the FROM and TO clips in a map clip to provide the editor with the most likely matching TO frames for each FROM frame selected by the editor for a roundabout transition.
130. A method according to claim 124, wherein the authoring and editing system further comprises means for using the mark in and mark out points of the FROM and TO clips in a map clip to provide the editor with the most likely matching TO frames for each FROM frame selected by the editor for a turn transition.
131. A method according to claim 123, wherein the authoring and editing system further comprises means for using the mark in and mark out points of the FROM and TO clips in a map clip to provide the editor with the most likely matching FROM frames for each TO frame selected by the editor for each requested roundabout transition.
132. A method according to claim 124, wherein the authoring and editing system further comprises means for using the mark in and mark out points of the FROM and TO clips in a map clip to provide the editor with the most likely matching FROM frames for each TO frame selected by the editor for each requested turn transition.
133. A method according to claim 124, wherein the authoring and editing system further comprises:
  - (a) means for using the mark in and mark out points of the FROM and TO clips and the relative lengths of the paths and the map positions of the paths to identify a probable intersection; and
  - (b) means for providing the editor with the most likely matching frames from the FROM and TO clips for each requested turn transition.

134. A method according to claim 123, wherein the authoring and editing system further comprises roundabout transition generating means whereby the authoring and editing system allows the user to preview a roundabout transition at edit time using the editor's provisionally selected FROM and TO frames.
135. A method according to claim 122, wherein the authoring and editing system further comprises turn transition generating means whereby the authoring and editing system allows the user to preview the turn transition at edit time using the editor's provisionally selected FROM and TO frames.
136. A computer controlled method of creating and editing hotspots in navigable motion video, comprising the steps of:
- (a) displaying a first frame of a video clip, which frame contains visual information representing a feature in the navigable motion video environment to be associated with a hotspot;
  - (b) drawing the hotspot on the displayed frame in response to user input so that the hotspot effectively encompasses the visual information representing the feature;
  - (c) creating a record of the identity, position and size of the hotspot and the identity of the first frame;
  - (d) displaying a second, non-adjacent frame of the video clip, which second frame also contains the visual information to be associated with a hotspot;
  - (e) drawing the hotspot on the displayed second frame in response to user input so that the hotspot substantially encompasses the visual information;
  - (f) creating a record of the identity, position and size of the hotspot and the identity of the second frame; and
  - (g) employing interpolation means to automatically draw the hotspot on all frames of the video clip between the first frame and the second frame in display order wherein the size values and position values of the hotspot drawn on each intermediate frame vary as a function of the proximity of the interpolated hotspot to each nearest user-drawn hotspot.

137. A method according to claim 136, wherein the steps of the method are carried out using a system which allows a plurality of users of the system to create and edit hotspots simultaneously.
138. A method according to claim 136, wherein the record of the identity, position and size of the hotspot drawn on the first frame and its association with the first frame, and the identity, position and size of the hotspot drawn on the second frame and its association with the second frame are stored in a database.
139. A method according to claim 136, further comprising the step of employing display mode means wherein the hotspot, when drawn, can be made visible on the display in response to user request.
140. A method according to claim 139, further comprising the step of employing editing means wherein the displayed hotspot can be altered in size or position in response to user input.
141. A method according to claim 139, wherein the hotspot display means further comprises display mode means for displaying interpolated hotspots at the request of the user.
142. A method according to claim 141, further comprising the steps of:
  - (a) employing means for playing the video clip as motion video or a single frame at a time with user drawn and interpolated hotspots displayed;
  - (b) employing means for stopping the video clip on any frame; and
  - (c) employing editing means wherein, in a change hotspot mode, the displayed hotspot, whether user-drawn or interpolated, can be altered in size or position in response to user input.

143. A method according to claim 142, further comprising the step of employing re-interpolation means wherein the interpolated size and position values of the hotspot in frames appearing between two user-drawn or user-altered frames are re-calculated and redrawn after the user draws or alters the hotspot in either such user-drawn or user-altered frame.
144. A method according to claim 136, wherein the interpolation is linear.
145. A method according to claim 143, wherein the interpolation is linear.
146. A method according to claim 141, wherein the hotspot appears as an outline in display mode.
147. A method according to claim 141, wherein, in display mode, edited hotspots appear as outlines of one color and interpolated hotspots appear as outlines of a different color.
148. A computer controlled method of creating and editing receptacles in navigable motion video, comprising the steps of:
- (a) displaying a first frame of a video clip, which frame contains visual information representing a feature in the navigable motion video environment to be associated with a receptacle,
  - (b) drawing a representation of the receptacle on the displayed first frame in response to user input so that the receptacle representation appears at the proper position and with the proper scale relative to the displayed feature;
  - (c) creating a record of the identity, position and scale factor of the receptacle as it appears in the first frame associated with the first frame;
  - (d) displaying a second, non-adjacent frame of the video clip, which second frame also contains the visual information representing a feature to be associated with a receptacle;

- (e) drawing a representation of the receptacle on the displayed second frame in response to user input so that the receptacle representation appears at the proper position and with the proper scale relative to the displayed feature;
  - (f) creating a record of the identity, position and scale factor of the receptacle associated with the first frame; and
  - (g) employing interpolation means to automatically draw representations of the receptacle on all frames of the video clip between the first frame and the second frame in display order wherein the scale factor and position values of the receptacle drawn on each between frame vary as a function of the proximity of the interpolated receptacle to each nearest user-drawn receptacle.
149. A method according to claim 148, wherein the steps of the method are carried out using a system which allows a plurality of users of the system to create and edit receptacles simultaneously.
150. A method according to claim 148, further comprising the steps of:
- (a) assigning an angle of view to the receptacle representation drawn on the first frame;
  - (b) assigning an angle of view to the receptacle representation drawn on the second frame; and
  - (c) employing interpolation means to automatically generate angles of view for the receptacle on all frames of the video clip between the first frame and the second frame in display order.
151. A method according to claim 148, wherein the record of the identity, position and scale factor of the receptacle drawn on the first frame and its association with the first frame, and the identity, position and scale of the receptacle drawn on the second frame and its association with the second frame are stored in a database.
152. A method according to claim 150, wherein a record of the identity, position, scale factor and angle of view of the receptacle drawn on the first frame and its association

with the first frame, and a record of the identity, position, scale factor and angle of view of the receptacle drawn on the second frame and its association with the second frame are stored in a database.

153. A method according to claim 148, further comprising the step of employing display mode means wherein the receptacle representation when drawn can be made visible on the display in response to user request.
154. A method according to claim 148, further comprising the step of assigning a sprite to the receptacle being edited or created.
155. A method according to claim 150, further comprising the step of displaying a default sprite in the receptacle in order to facilitate editing of the position and scale factor of the receptacle.
156. A method according to claim 150, further comprising the step of displaying a default multiple view sprite in the receptacle in order to facilitate editing of the position, scale factor, and angle of view of the receptacle.
157. A method according to claim 150, further comprising the steps of:
  - (a) assigning a sprite with multiple angle of view images to the receptacle being edited or created;
  - (b) selecting an angle of view for the first frame and viewing the sprite face corresponding to that angle of view as drawn in the receptacle;
  - (c) adjusting the angle of view if needed and viewing the sprite face corresponding to the adjusted angle of view;
  - (d) selecting an angle of view for the second frame and viewing the sprite face corresponding to that angle of view as drawn in the receptacle;
  - (e) adjusting the angle of view if needed and viewing the sprite face corresponding to the adjusted angle of view.



158. A method according to claim 157, further comprising the steps of:
- (a) employing means for playing the video clip as motion video or a single frame at a time with the sprite face corresponding to the user selected and interpolated angles of view displayed;
  - (b) employing means for stopping the video clip on any frame; and
  - (c) employing editing means wherein, in a change receptacle mode, the displayed receptacle, whether user-drawn or interpolated, can be altered in scale factor, position or angle of view in response to user input.
159. A method according to claim 158, further comprising the step of employing re-interpolation means wherein the interpolated scale factor, position values and angles of view for the receptacle in frames appearing between two user-drawn or user-altered frames are re-calculated and redrawn after the user draws or alters one of the receptacle representations in either such user-drawn or user-altered frame.
160. A method according to claim 159, wherein the interpolation is linear.
161. A method according to claim 159, further comprising means for selecting and assigning to the receptacle the closest available angle of view to the interpolated angle of view for frames where there is no face available corresponding to the interpolated angle of view.
162. A method according to claim 159, wherein records of the receptacle identity, frame specific interpolated scale factors, angles of view and position data are stored in a database.
163. A method according to claim 153, further comprising the step of employing editing means wherein the position and scale factor of a displayed receptacle representation can be altered in response to user input.

164. A method according to claim 153, wherein the receptacle display means further comprises display mode means for displaying interpolated receptacles at the request of the user.
165. A method according to claim 169, wherein the receptacle display means further comprises display mode means for displaying interpolated receptacles and user-drawn or user-altered receptacles in contrasting colors.
166. A method according to claim 164, further comprising the steps of:
- (a) employing means for playing the video clip as motion video or a single frame at a time with user drawn and interpolated receptacles displayed;
  - (b) employing means for stopping the video clip on any frame; and
  - (c) employing editing means wherein, in a change receptacle mode, the displayed receptacle, whether user-drawn or interpolated, can be altered in scale factor or position in response to user input.
167. A method according to claim 166, further comprising the step of employing re-interpolation means wherein the interpolated scale factor and position values of the receptacle in frames appearing between two user-drawn or user-altered frames are recalculated and redrawn after the user draws or alters one of the receptacle representations in either such user-drawn or user-altered frame.
168. A method according to claim 148, wherein the interpolation is linear.
169. A method according to claim 166, wherein the interpolation is linear.
170. A method according to claim 164, wherein the receptacle appears as an outline in display mode.
171. A method according to claim 166, wherein records of the receptacle identity and frame specific interpolated scale factors and position data are stored in a database.

172. A method according to claim 148, further comprising the step of selecting an anchor point to be associated with the receptacle.
173. A method according to claim 172, further comprising the steps of:
- (a) selecting a sprite to be associated with the receptacle;
  - (b) viewing the sprite overlayed on the receptacle; and
  - (c) adjusting the anchor point if needed.
174. A method according to claim 150, further comprising the steps of:
- (a) associating a sprite with multiple angle of view images with the receptacle;
  - (b) employing means for playing the video clip as motion video or a single frame at a time with the user drawn and interpolated receptacles containing the user selected or interpolated angle of view face of the sprite displayed;
  - (b) employing means for stopping the video clip on any frame; and
  - (c) employing editing means wherein, in a change receptacle mode, the displayed receptacle, whether user-drawn or interpolated, can be altered in scale factor, position or angle of view in response to user input.
175. An M-view data structure for insertion into a navigable motion video production, comprising:
- (a) at least one M-view object;
  - (b) at least one M-view face associated with each M-view object;
  - (c) at least one angle associated with each M-view face; and
  - (d) a size number associated with each M-view object.
176. An M-view according to claim 175, wherein the M-view is associated with at least one receptacle in a navigable motion video production.
177. An M-view according to claim 175, wherein a record of the M-view, its associated receptacle, angle and size number is stored in a database.

178. An M-view according to claim 175, further comprising a plurality of animation states associated with each M-view face.
179. An M-view according to claim 178, further comprising a program flow associated with the M-view object, wherein the appropriate animation state and the appropriate face of the M-view are displayed.
180. A method for the creation and insertion of a sprite in a navigable motion video, comprising the step of creating at least one image of an object to be represented by the sprite.
181. A method according to claim 180, wherein said sprite is animated.
182. A method according to claim 180, wherein said step of creating at least one image of an object comprises the substep of creating a series of images of the faces of the object.
183. A method according to claim 182, wherein said faces represent different angles of view of the object as it is rotated step-wise around at least one axis of rotation.
184. A method according to claim 182, further comprising the step of creating a series of images of different animation states of the faces.
185. A method according to claim 183, further comprising the steps of:
  - (a) importing the images into a computer-based data storage system; and
  - (b) using a database to associate angle information with the stored images.
186. A method according to claim 183, further comprising the step of storing the images as bitmaps in a data storage system.

187. A method according to claim 183, further comprising the step of storing the images as digitally compressed images in a data storage system.
188. A method according to claim 183, wherein between 1 and 360 faces are created for each axis of rotation used for a particular object.
189. A method according to claim 183, further comprising the step of using a computer to select the optimal number of faces and angles for the sprite.
190. A method according to claim 183, further comprising the step of assigning a size number to the sprite.
191. A method according to claim 183, further comprising the step of assigning a resolution to the sprite.
192. A method according to claim 191, further comprising the step of assigning a plurality of resolutions to the sprite, whereby the sprite is initially rendered in the production at the lower resolution and updated to a higher resolution when computing resources are available to do so.
193. A method according to claim 192, wherein the lower resolution images of the sprite are stored in a memory buffer at runtime to provide for quick rendering at runtime.
194. A method according to claim 192, wherein the lower resolution images of the sprite are stored in a memory buffer at runtime as digitally compressed images which are decompressed on the fly immediately prior to being rendered into a display buffer.
195. A method according to claim 183, further comprising the steps of:
  - (a) assigning the sprite to a receptacle in a navigable motion video clip;
  - (b) using a navigable motion video authoring and editing system to access the first frame of a navigable motion video clip where the receptacle appears;

- (c) selecting a face for the sprite to be associated with the first frame to create the appropriate perspective view of the sprite;
  - (d) using the authoring and editing system to access the last frame of the video clip where the receptacle appears;
  - (e) selecting a face for the sprite to be associated with the last frame to maintain the appropriate viewer perspective;
  - (f) assigning a face to the frames between the first frame and last frame using interpolation between the face angle of the first frame and the face angle of the last frame to maintain the appropriate perspective; and
  - (g) where the angle required by the interpolation is not available, choosing the closest matching angle.
196. A method according to claim 195, further comprising the step of assigning a animation flow to the sprite which selects which animation state of the face to display and selects the speed of animation.
197. A method according to claim 195, wherein the assigning of faces to the frames between the first frame and last frame is accomplished by computer-implemented linear interpolation.
198. A method according to claim 195, wherein the selection of the closest available face is accomplished by a computer implemented comparison of available face angles with the interpolated angle value.
199. A method according to claim 183, further comprising the step of storing the sprite images as digitally compressed bitmaps.
200. A method according to claim 183, further comprising the step of rendering the bitmap image of the sprite onto a display buffer in which the bitmap of the frame on which the sprite is to be rendered has already been placed.

201. A method according to claim 200, wherein the transparent pixels of the sprite bitmap are not copied into the display buffer.
202. A method according to claim 200, wherein anti-aliasing means are used to smooth the edge of the sprite bitmap.
203. A method according to claim 190, further comprising the step of sizing the sprite bitmap to be copied onto a particular frame according to the size number of the sprite and the actual size of the receptacle.
204. A method according to claim 203, wherein the sizing of the sprite bitmap is accomplished using a fast algorithm.
205. A method according to claim 204, wherein:
  - (a) the sprite bitmap is stored in a digitally compressed format;
  - (b) the bitmap is stretched by duplicating approximately evenly spaced rows and columns; and
  - (c) the bit map is shrunk by deleting approximately evenly spaced rows and columns.
206. A method according to claim 200, wherein the display of navigable motion video containing a sprite is updated by means of a double or triple buffer wherein the video frame image and sprite image are constructed in at least one memory buffer and the completed, combined image is copied into a display buffer before being displayed.
207. A method according to claim 183, wherein the object faces are created from computer models.
208. A method according to claim 183, wherein the object faces are created from digital images of a real object.

209. A method according to claim 195, further comprising the step of, at the time a particular video clip is accessed during runtime of a motion video production, loading the faces of sprites which are present in the clip into memory for all sprites appearing in the accessed video clip.
210. A method according to claim 209, wherein said step of loading the faces of sprites includes loading the animation states for each face.
211. A method according to claim 195, wherein only the faces actually used in a video production are stored with the bound production.
212. A method according to claim 211, wherein only the faces actually used in a video production are stored with the bound production.
213. A method of binding a navigable motion video production, comprising the steps of:
- (a) creating a data hunk on a computer-readable medium;
  - (b) creating a logic hunk on the computer-readable medium; and
  - (c) creating a clip hunk on the computer-readable medium.
214. A method according to claim 213, wherein said data hunk, said logic hunk, and said clip hunk contain only information which is absolutely necessary for execution of the production at runtime.
215. A method according to claim 213, wherein said data hunk comprises information needed to set up a feature selected from the group consisting of hotspots, receptacles, visuals, global properties, music, and asynchronous inputs.
216. A method according to claim 213, wherein said logic hunk contains program logic.
217. A method according to claim 213, wherein said clip hunk contains video data.



218. A method according to claim 217, wherein said video data is digitally compressed.
219. A method according to claim 217, wherein said video data is in an MPEG format.
220. A method according to claim 213, further comprising the step of creating an initialization file on the computer-readable medium, said initialization file containing information about the location of files on the computer-readable medium.
221. A method according to claim 213, further comprising a relocation step wherein any reference in one hunk to an item in another hunk is replaced by a numerical offset reference.
222. A method of binding a navigable motion video production, comprising the steps of:
- (a) determining which information associated with the production is absolutely necessary for execution of the production at runtime; and
  - (b) storing on a computer-readable medium only the information which is absolutely necessary for execution of the production at runtime, thereby conserving storage space.
223. A navigable multimedia system, comprising a computer programmed to:
- (a) display a navigable motion video-based representation of an environment having paths, wherein each of said paths is depicted by at least one video clip; and to
  - (b) allow for spontaneous navigation of the paths of the video-based representation of the environment in response to user input.
224. A navigable multimedia system as recited in claim 223, wherein said video clip is stored in a digitally compressed format.
225. A navigable multimedia system as recited in claim 224, wherein said video clip is stored in an MPEG format.

226. A navigable multimedia system as recited in claim 223, wherein:
- (a) said video clip has a field of view greater than the field of view displayed at any one time; and
  - (b) said computer is further programmed to pan the display of the video-based representation of the environment to the undisplayed portion of the video clip in response to user input, thereby creating the visual impression of a turn.
227. A navigable multimedia system as recited in claim 223, wherein:
- (a) at least one of said paths is depicted by a plurality of video clips representing a plurality of camera headings along the path; and
  - (b) said computer is further programmed to execute a transition from one clip depicting the path to another clip depicting the same path from a different camera heading in response to user input, thereby creating the visual impression of a rotation of view on a single path.
228. A navigable multimedia system as recited in claim 227, wherein said transition creates the visual impression of a seamless rotation of view.
229. A navigable multimedia system as recited in claim 227, wherein:
- (a) said transition is executed between a FROM frame in one video clip and a TO frame in the other video clip, each of which depicts the same point on the path from a different camera heading; and
  - (b) said computer is further programmed such that, when the user begins navigating the path by playing one video clip associated with the path, the TO frames for points along the path at which transitions can occur are loaded into said computer's memory.
230. A navigable multimedia system as recited in claim 227, wherein said plurality of video clips together form a 360° view of the path.

231. A navigable multimedia system as recited in claim 227, wherein said plurality of video clips each have a field of view of about 90° and are directionally offset from the adjacent clip by about 90°.
232. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to execute a transition from a clip depicting the path currently being navigated to a clip depicting an intersecting path in response to user input at the point of intersection of the two paths, thereby creating the visual impression of a seamless turn from one path in the environment onto another.
233. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to display a bitmap representation of an object in proper perspective along with said motion video representation of the environment, thereby creating the impression that the object is present in the environment.
234. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to associate hotspots with certain portions of the motion video representation of the environment, said hotspots serving as a means for user input.
235. A navigable multimedia system as recited in claim 234, wherein said computer is further programmed to execute a flow in response to user interaction with a hotspot.
236. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to play dramatic video clips when the user navigating the representation of the environment provides the proper input in the portions of the representation of the environment associated with said dramatic video clips.
237. A navigable multimedia system as recited in claim 236, wherein said dramatic video clips serve as non-navigable video connectors between different areas of the environment.

238. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to play audio clips as the user navigates through portions of the representation of the environment associated with said audio clips.
239. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to display a two-dimensional map of the environment.
240. A navigable multimedia system as recited in claim 223, wherein said computer is further programmed to display visuals simultaneously with the navigable motion video-based representation of the environment.
241. A computer readable medium for causing a computer to display a navigable motion video-based representation of an environment, comprising:
- a. a computer-readable storage medium; and
  - b. a computer program stored on said storage medium, said computer program comprising:
    - i. means for causing a computer to display a video-based representation of an environment having paths, wherein each of said paths is depicted by at least one video clip; and
    - ii. means for enabling a user to spontaneously navigate the paths of the video-based representation of the environment.
242. A computer readable medium as recited in claim 241, wherein said computer program is interpreter-based.
243. A computer readable medium as recited in claim 242, wherein said computer program further comprises:
- a. a flow interpreter for managing flow logic during program execution;
  - b. a video interpreter for managing the playing of the video clips during program execution.

244. A computer readable medium as recited in claim 241, wherein said computer program is compiled.
245. A system for tracking the position of an object within a tracking area, comprising:
- a. an ultrasonic transmitter for generating an ultrasonic pulse, said transmitter mounted on the object to be tracked;
  - b. a plurality of ultrasonic receivers located at different locations in the tracking area for receiving the ultrasonic pulse;
  - c. means for measuring the time delays from the time at which the ultrasonic pulse is generated until the ultrasonic pulse is received by each of the ultrasonic receivers; and
  - d. means for calculating the position of the object based upon the measured time delays, which means includes a computer programmed to execute a push-pull positioning algorithm to determine the position of the object.
246. A system for tracking the position of an object as recited in claim 245, further comprising an electronic compass and an inclinometer mounted on the object to be tracked for measuring the azimuth, pitch, and roll of the object.
247. A method of tracking the position of an object within a tracking area, comprising the steps of:
- a. generating an ultrasonic pulse from the location of the object to be tracked;
  - b. measuring the time delays between the generation of the ultrasonic pulse and its receipt at a plurality of locations in the tracking area; and
  - c. calculating the position of the object using a push-pull positioning algorithm based upon the measured time delays.

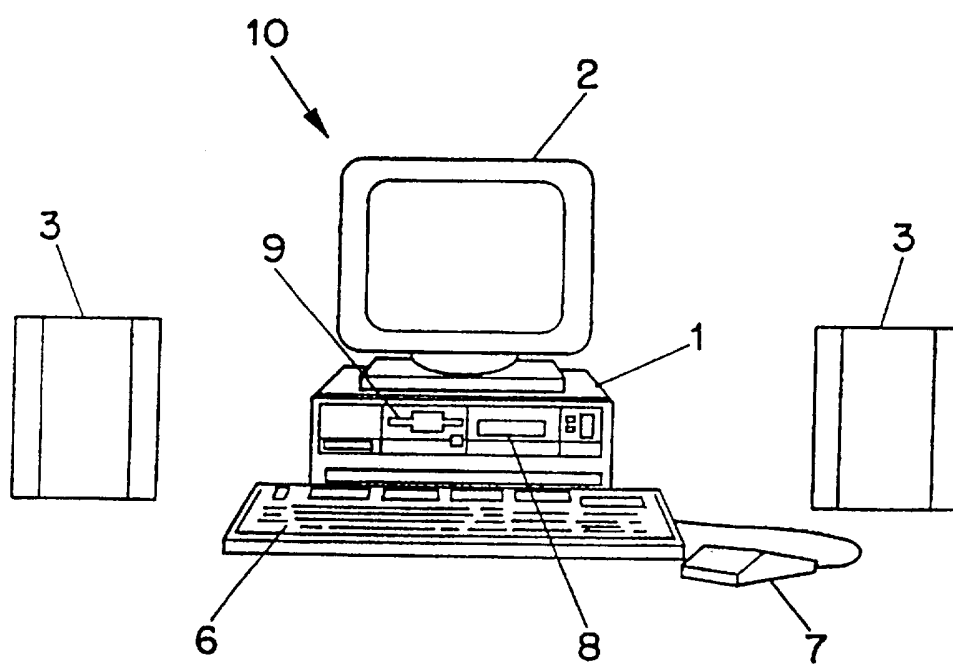


Fig. 1

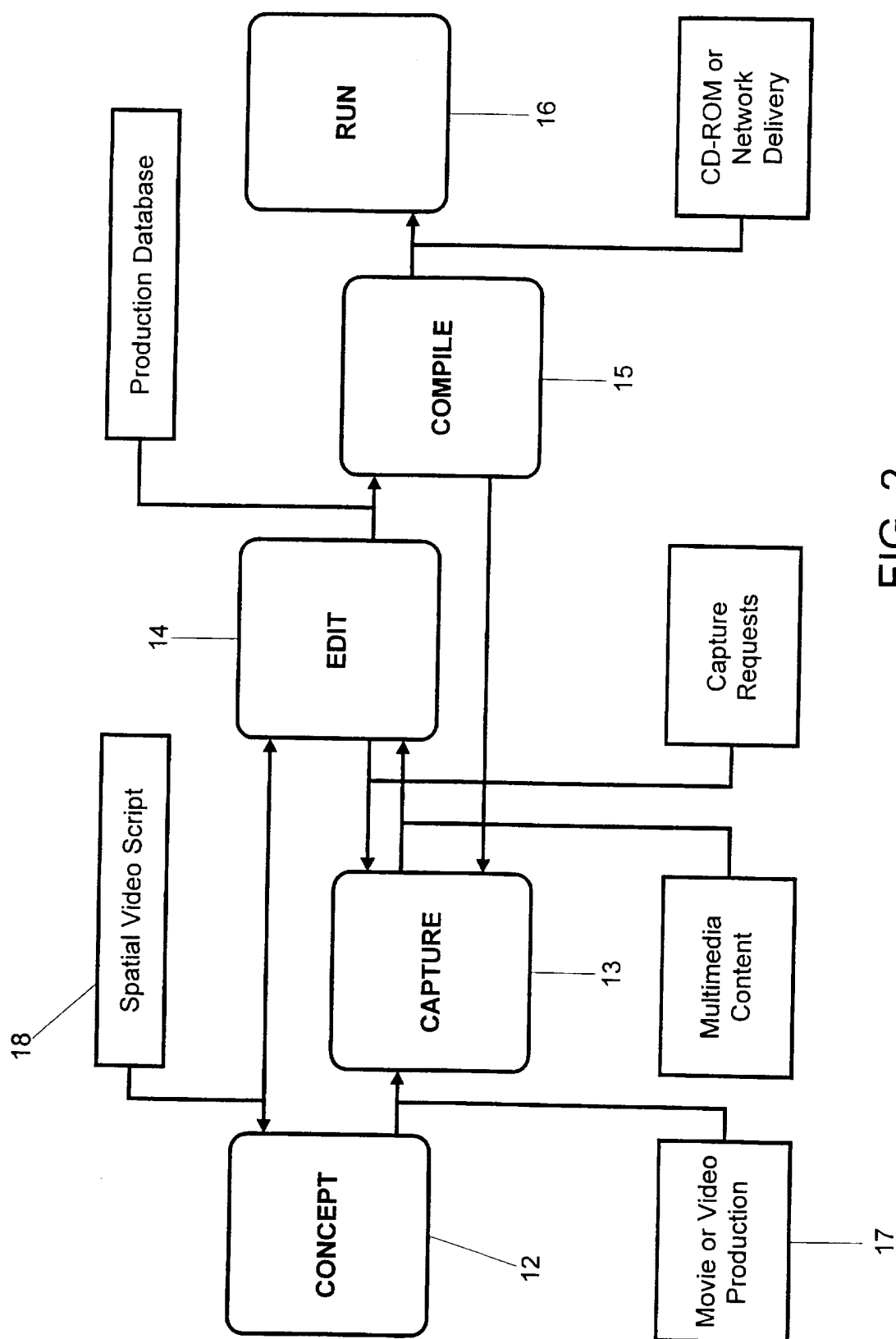


FIG. 2  
Spatial Video Production Process

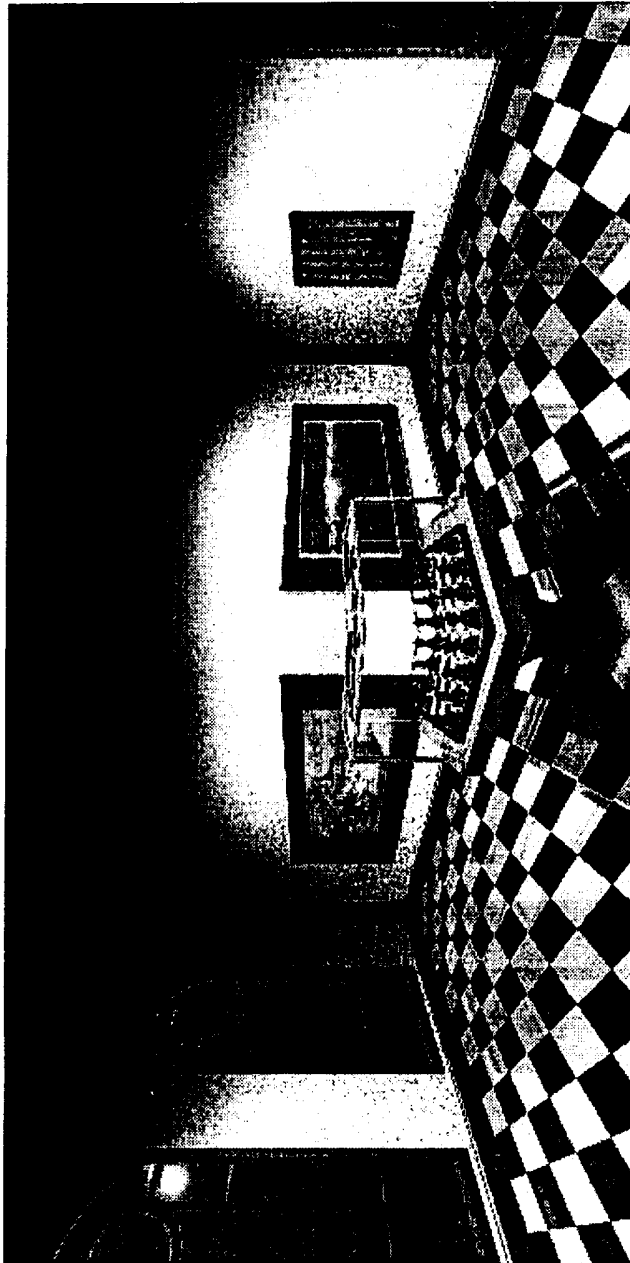


FIG. 3A



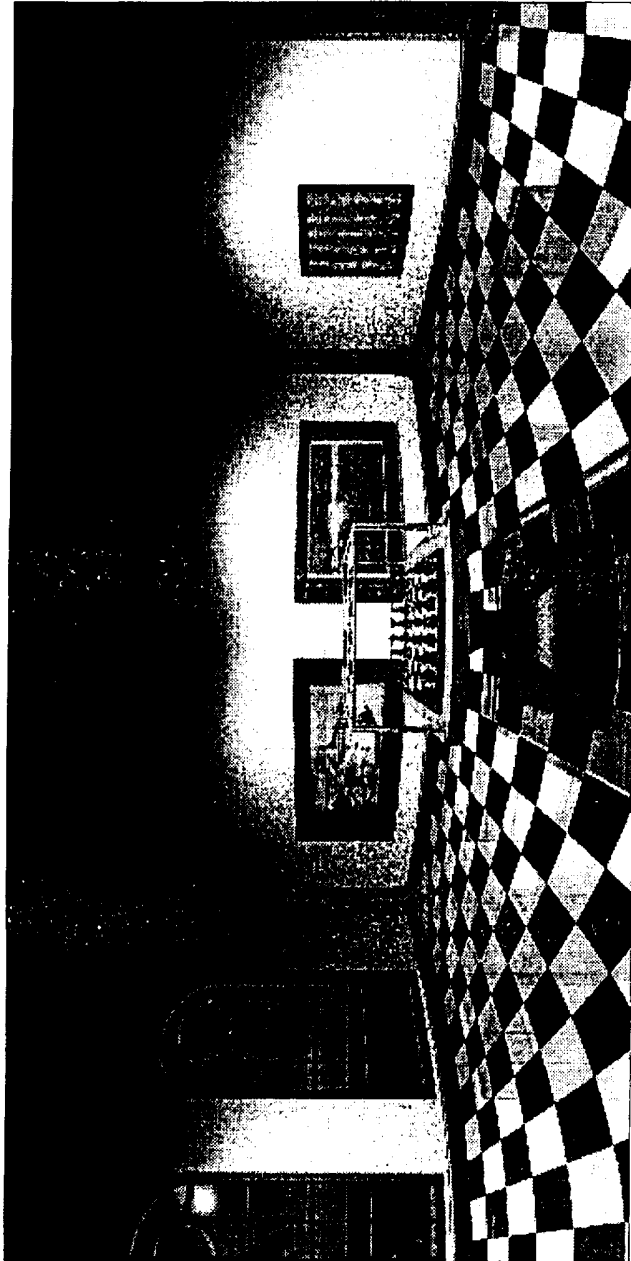


FIG. 3B

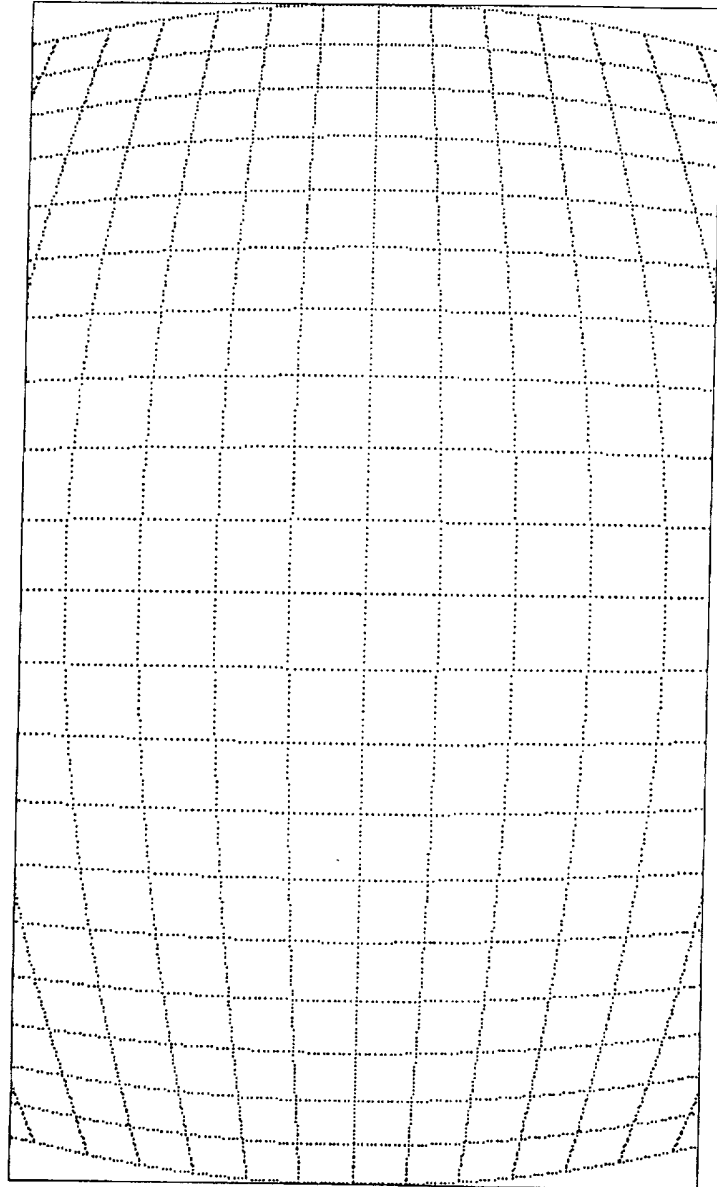


FIG. 4A

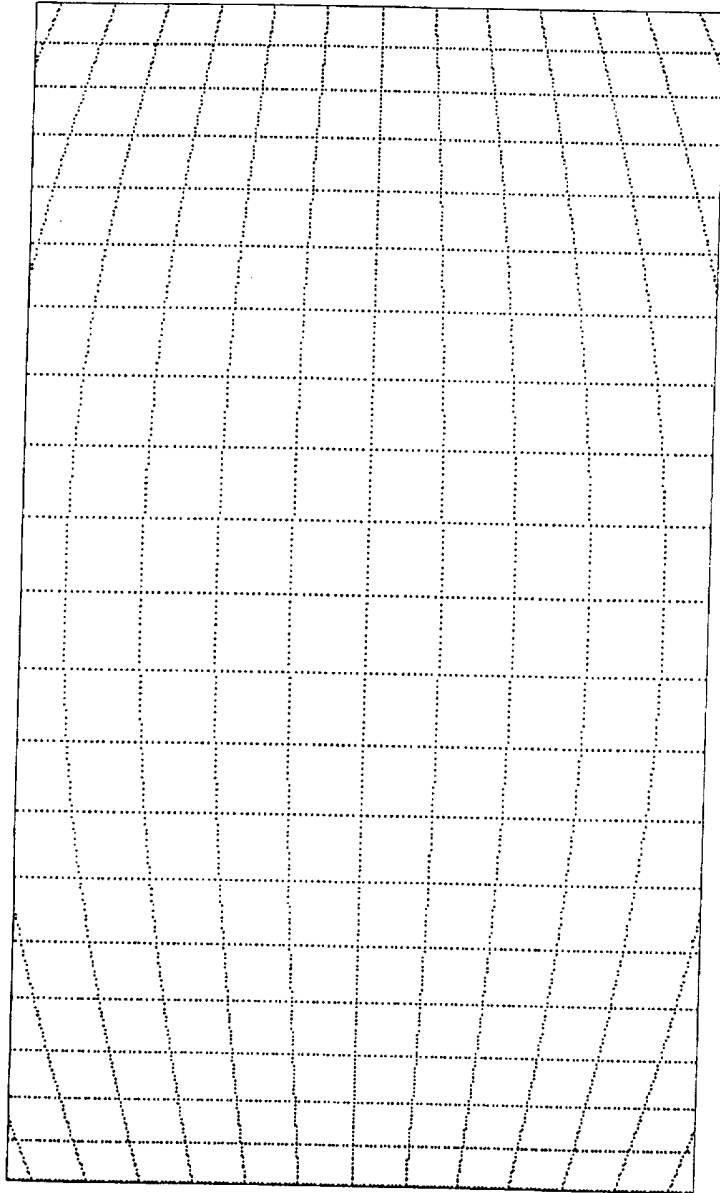
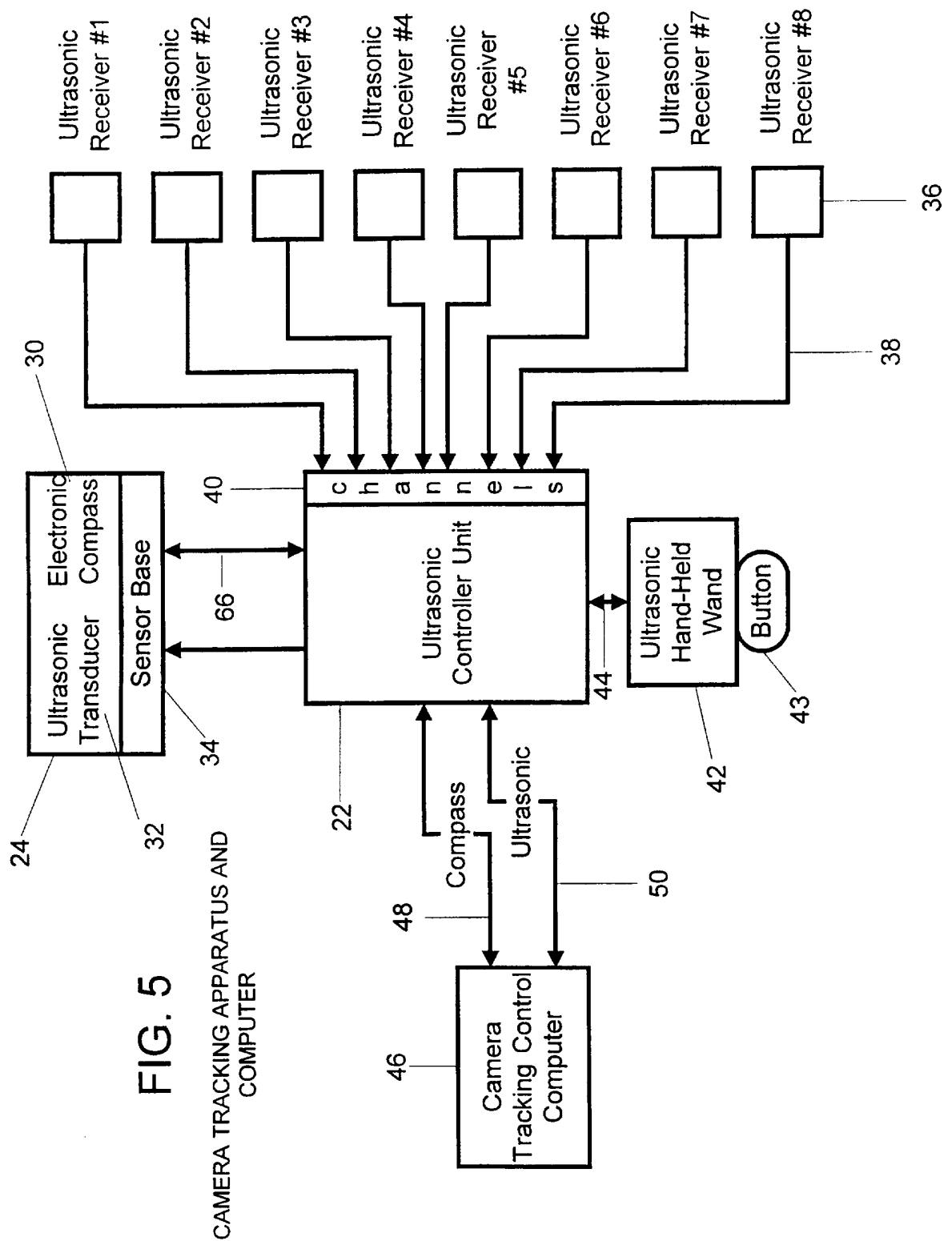
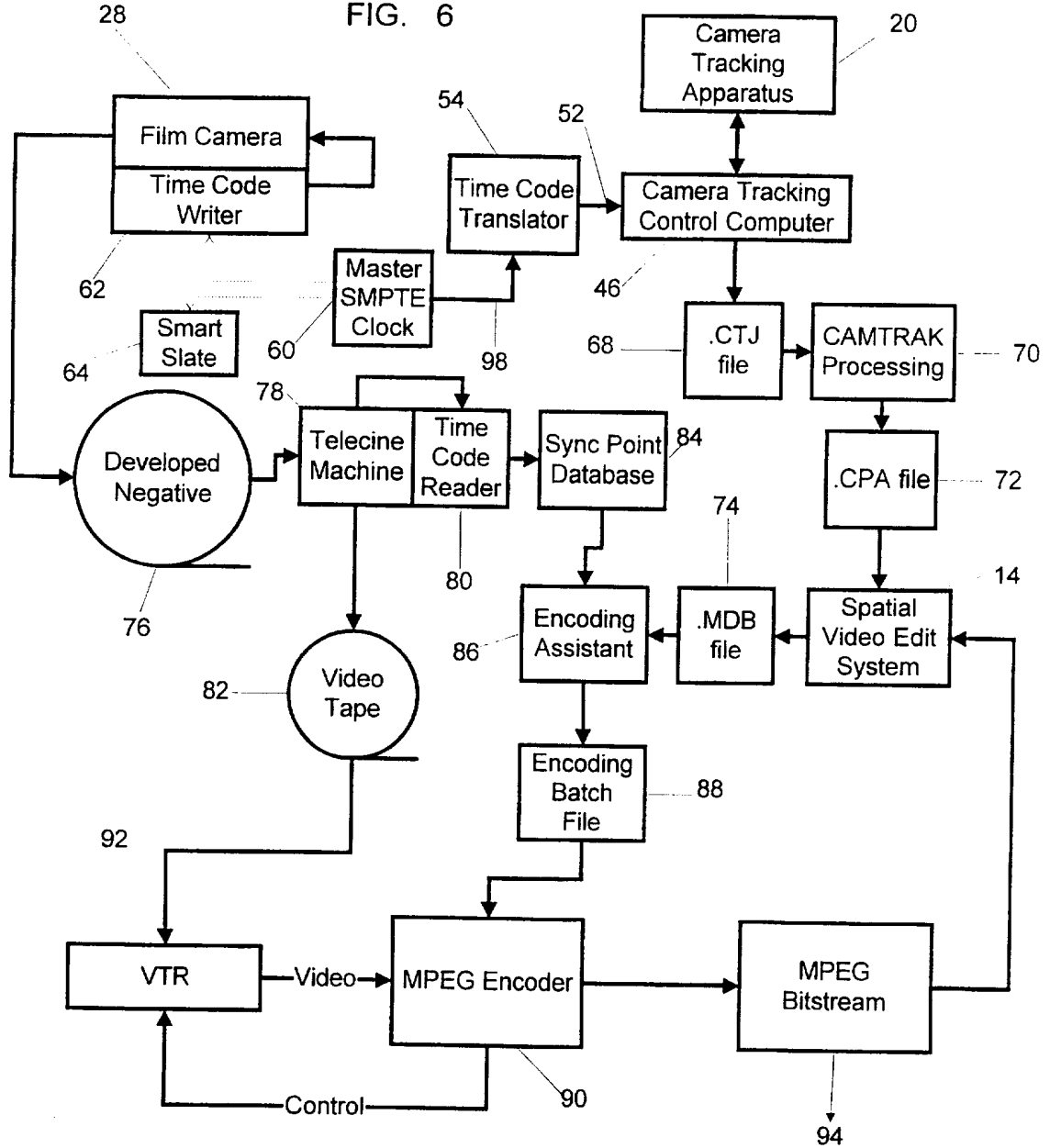
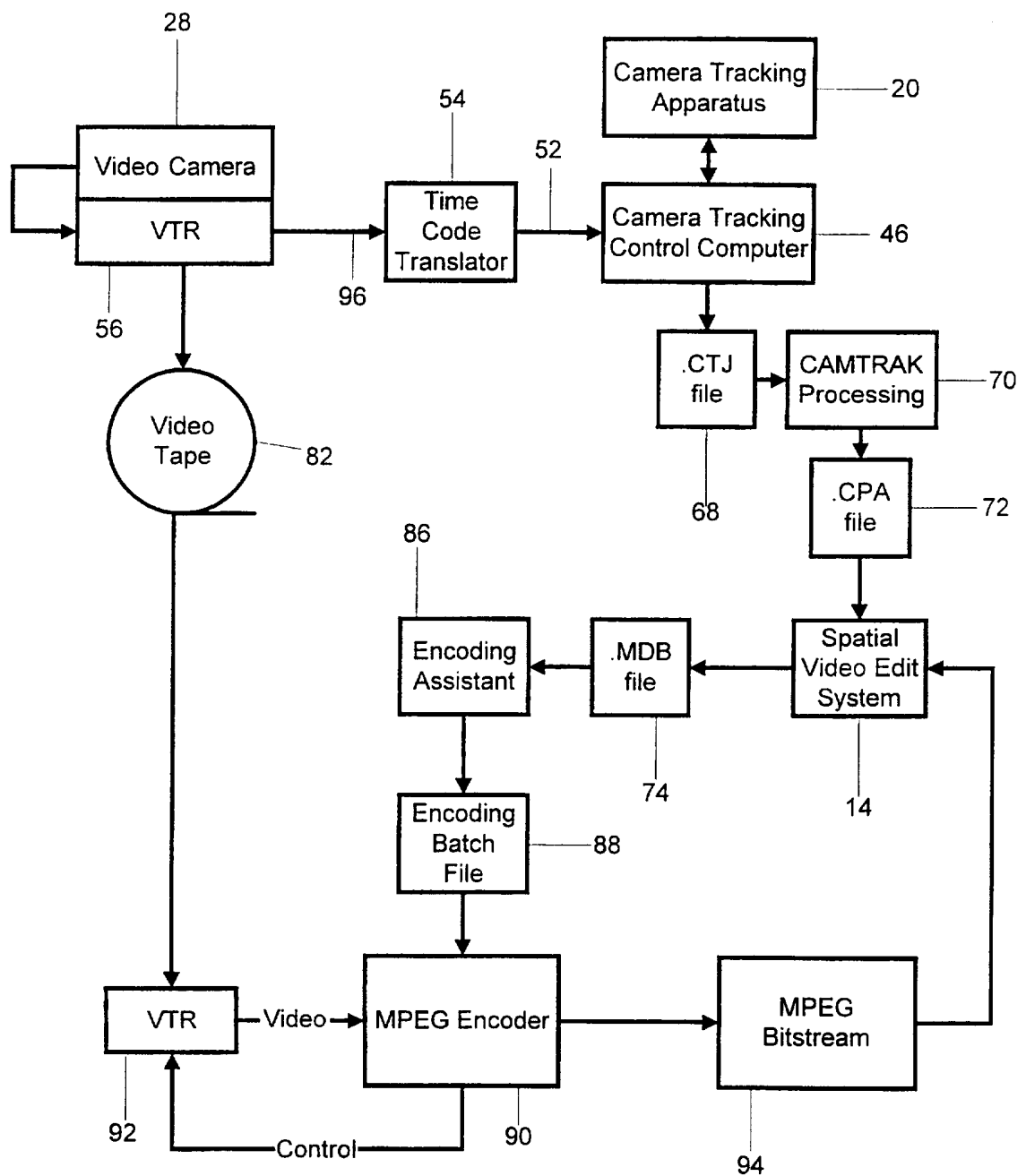


FIG. 4B



CAPTURE - FILM  
FIG. 6



CAPTURE-VIDEO  
FIG. 7

FIG. 8

Push-Pull Process

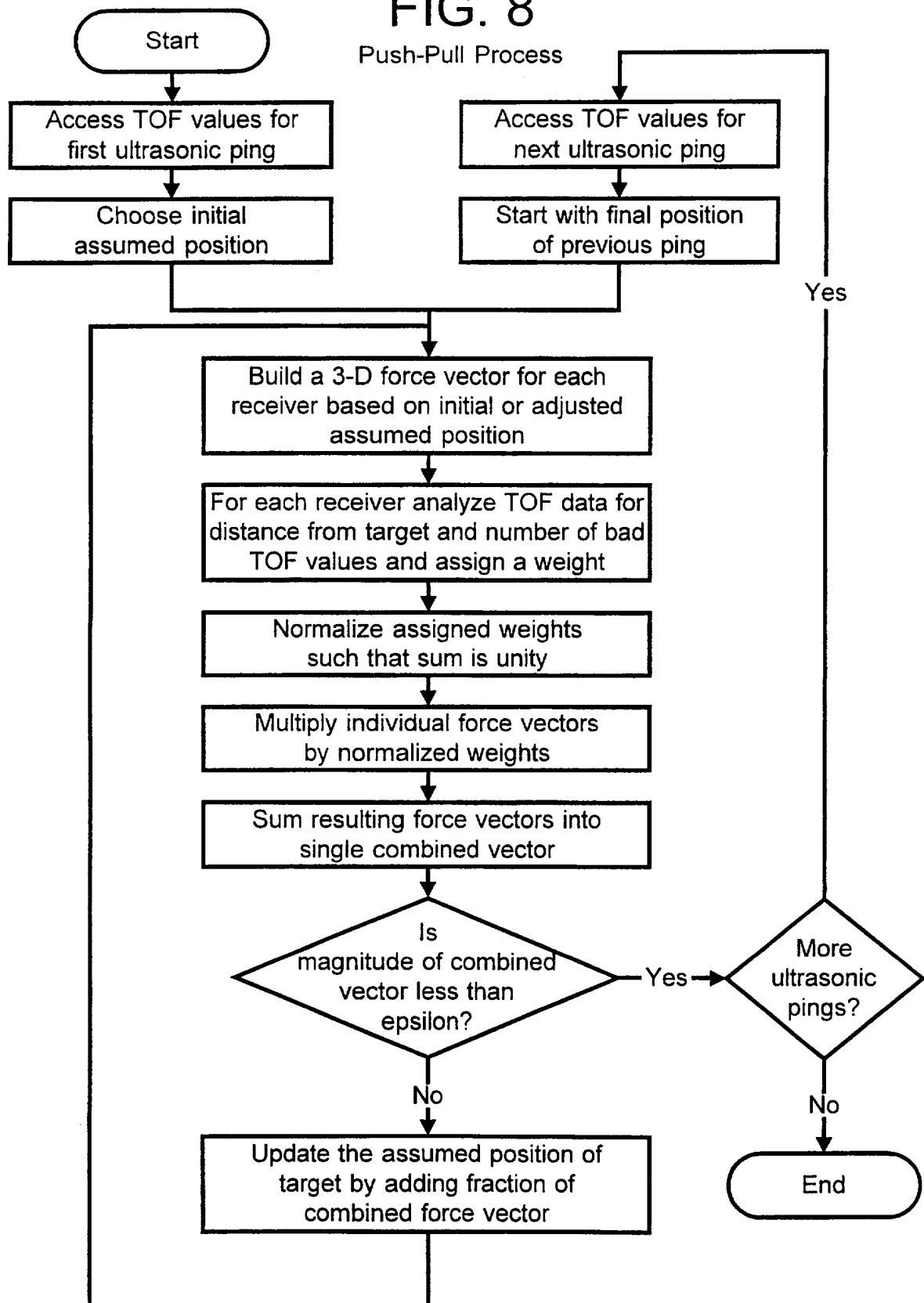
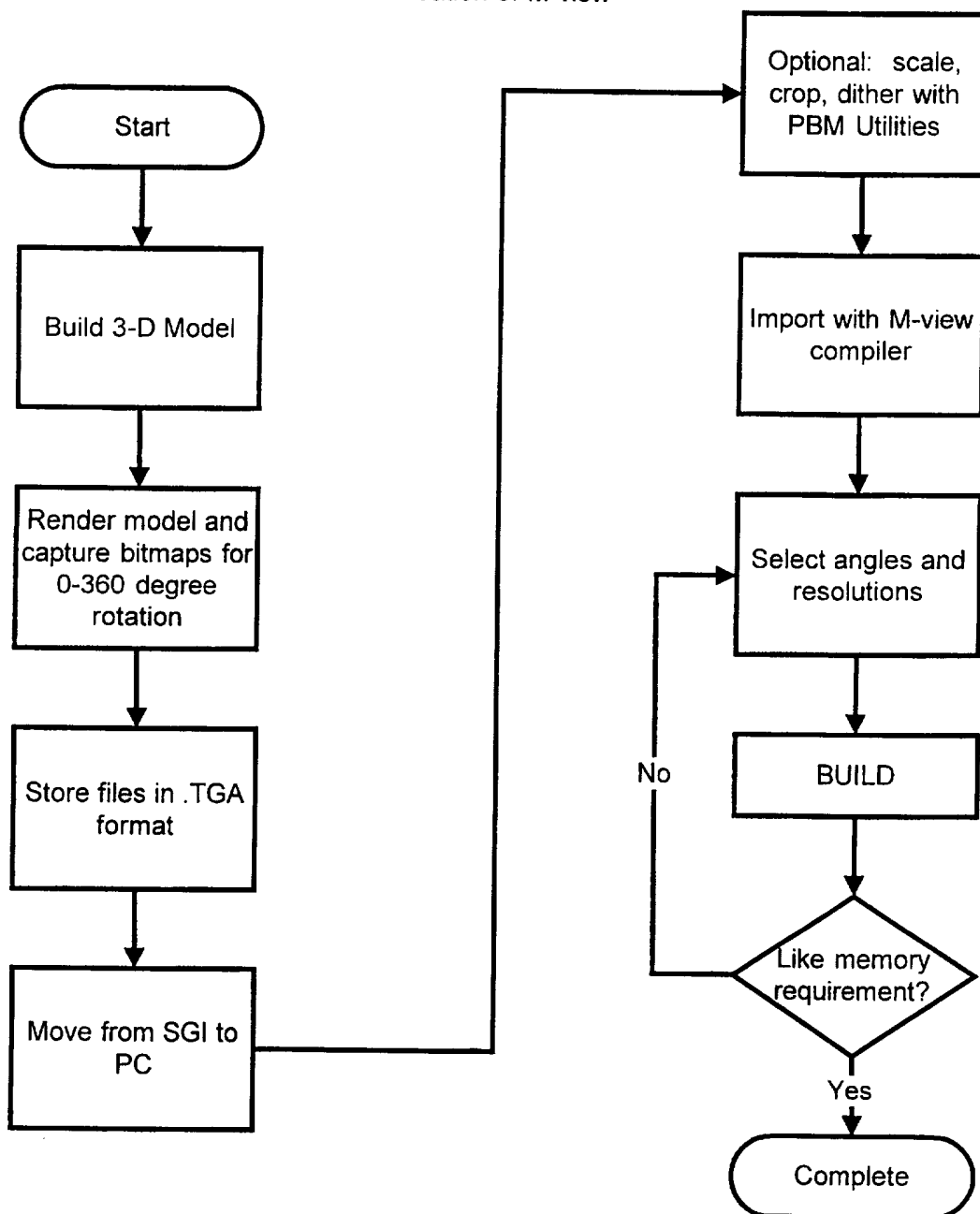


FIG. 9  
Creation of M-view





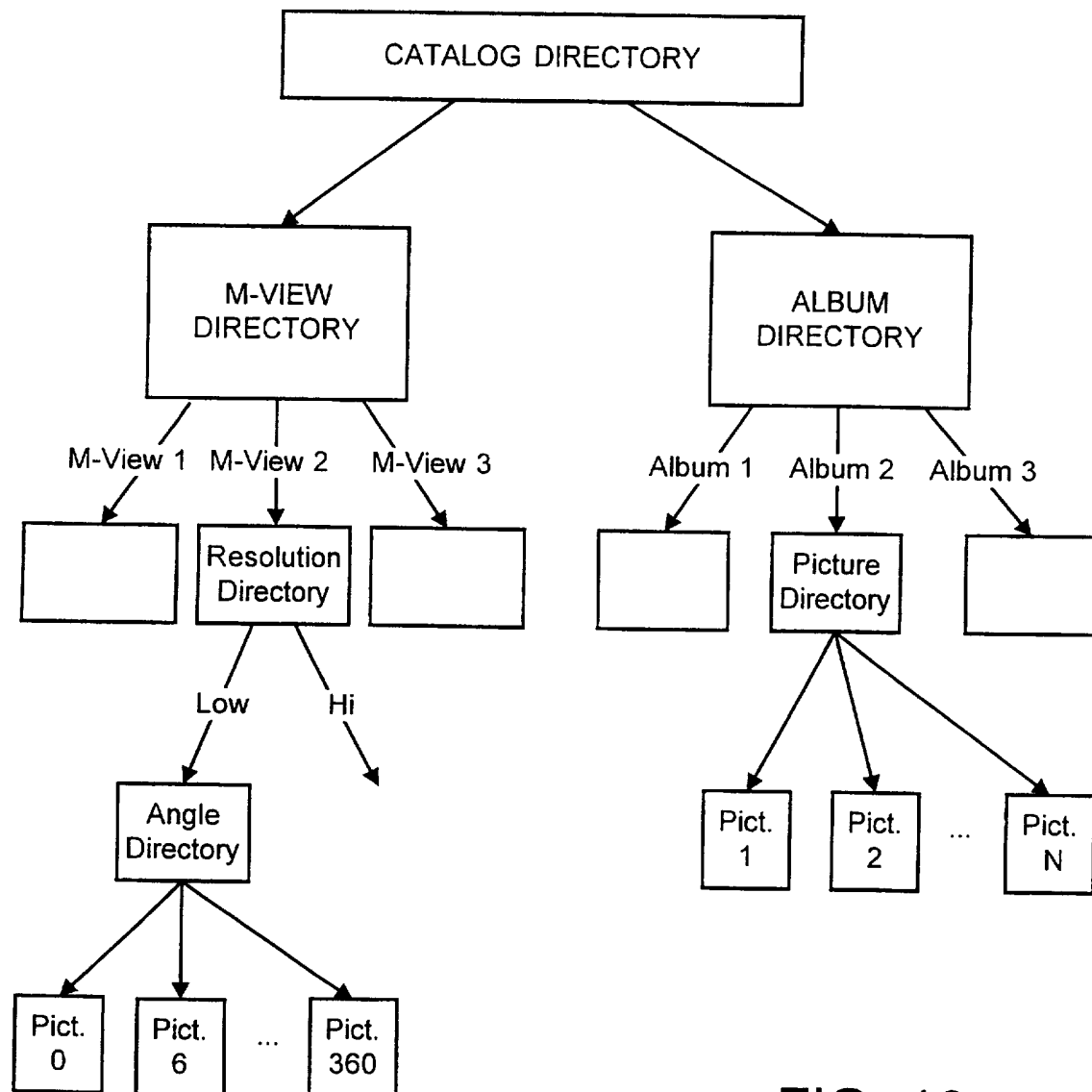
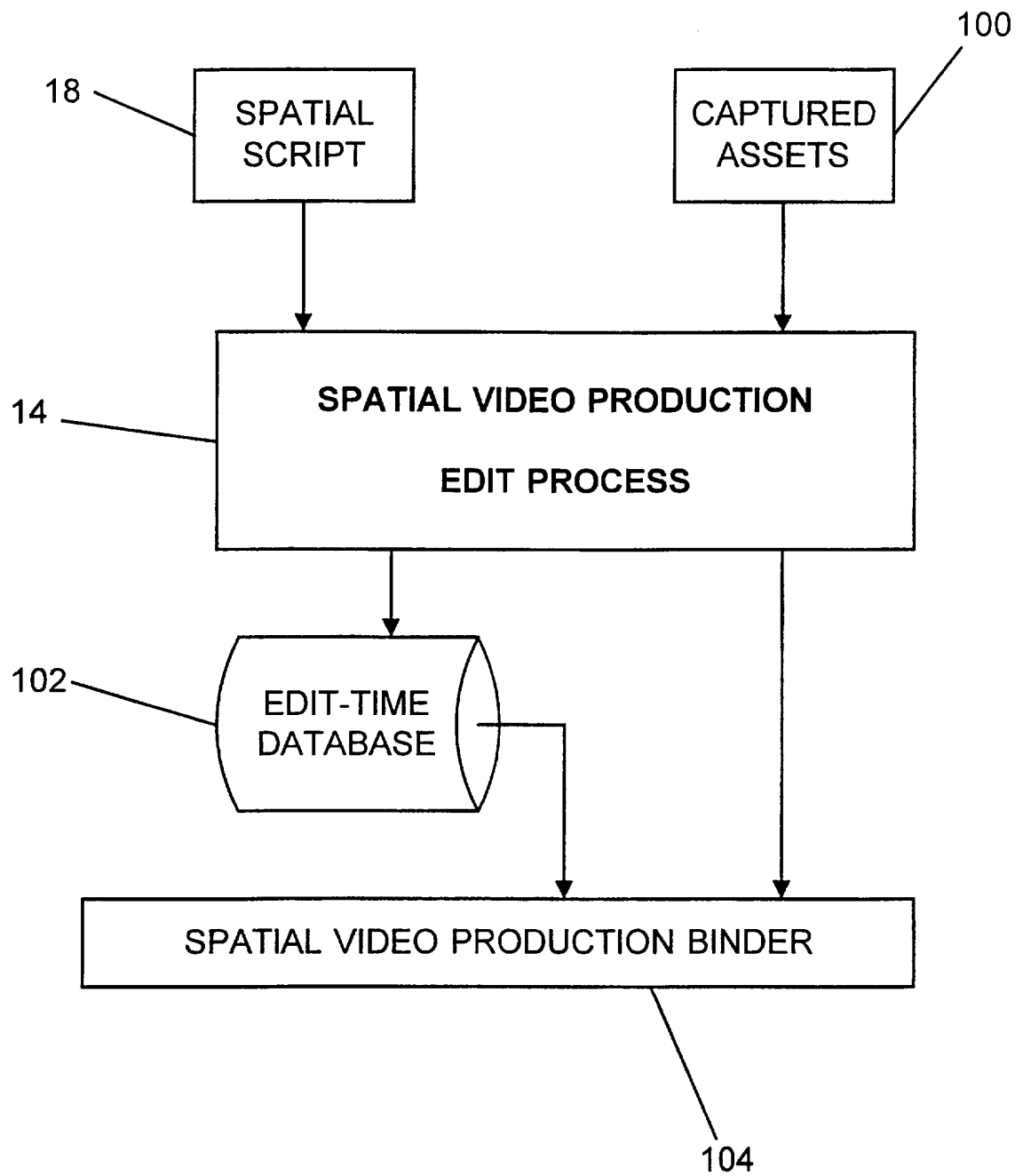
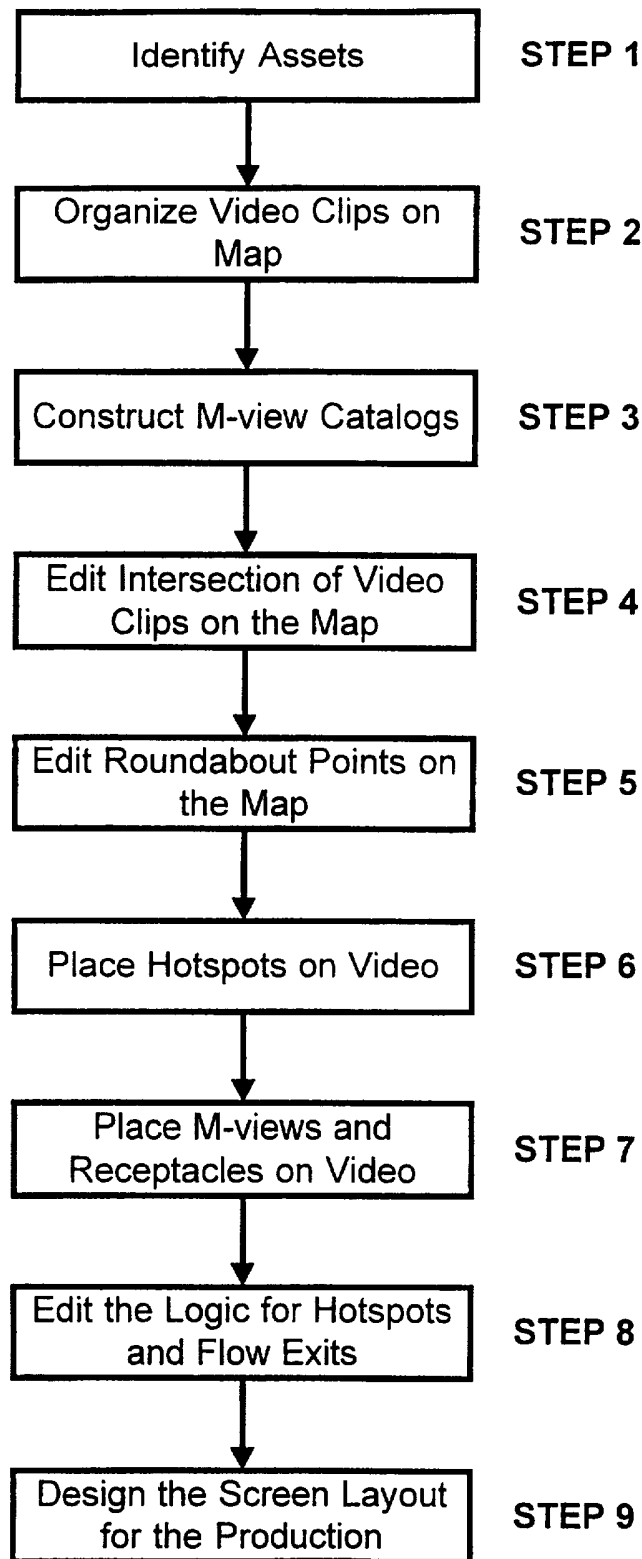


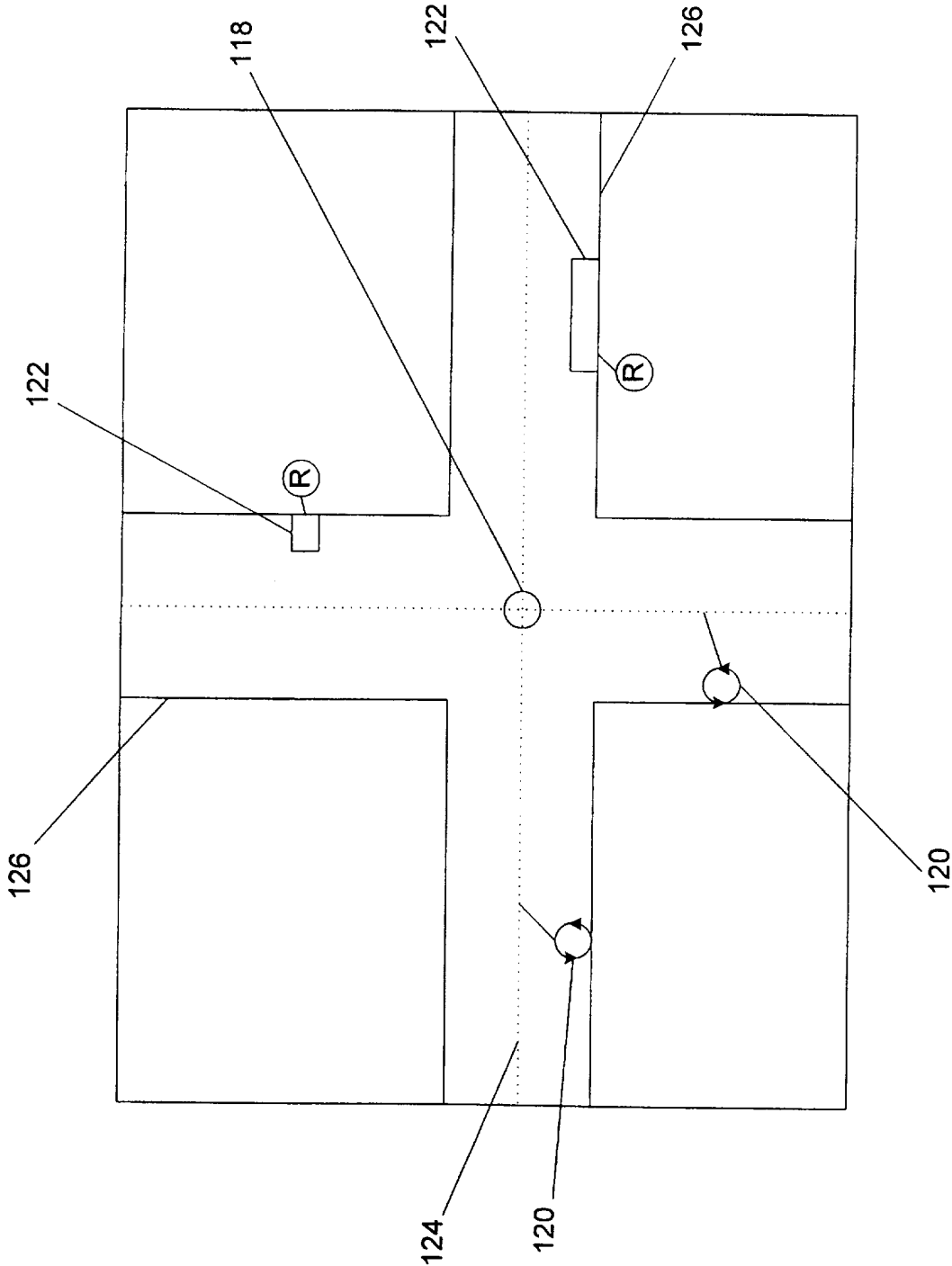
FIG. 10

M-view Catalog Format

**FIG. 11**



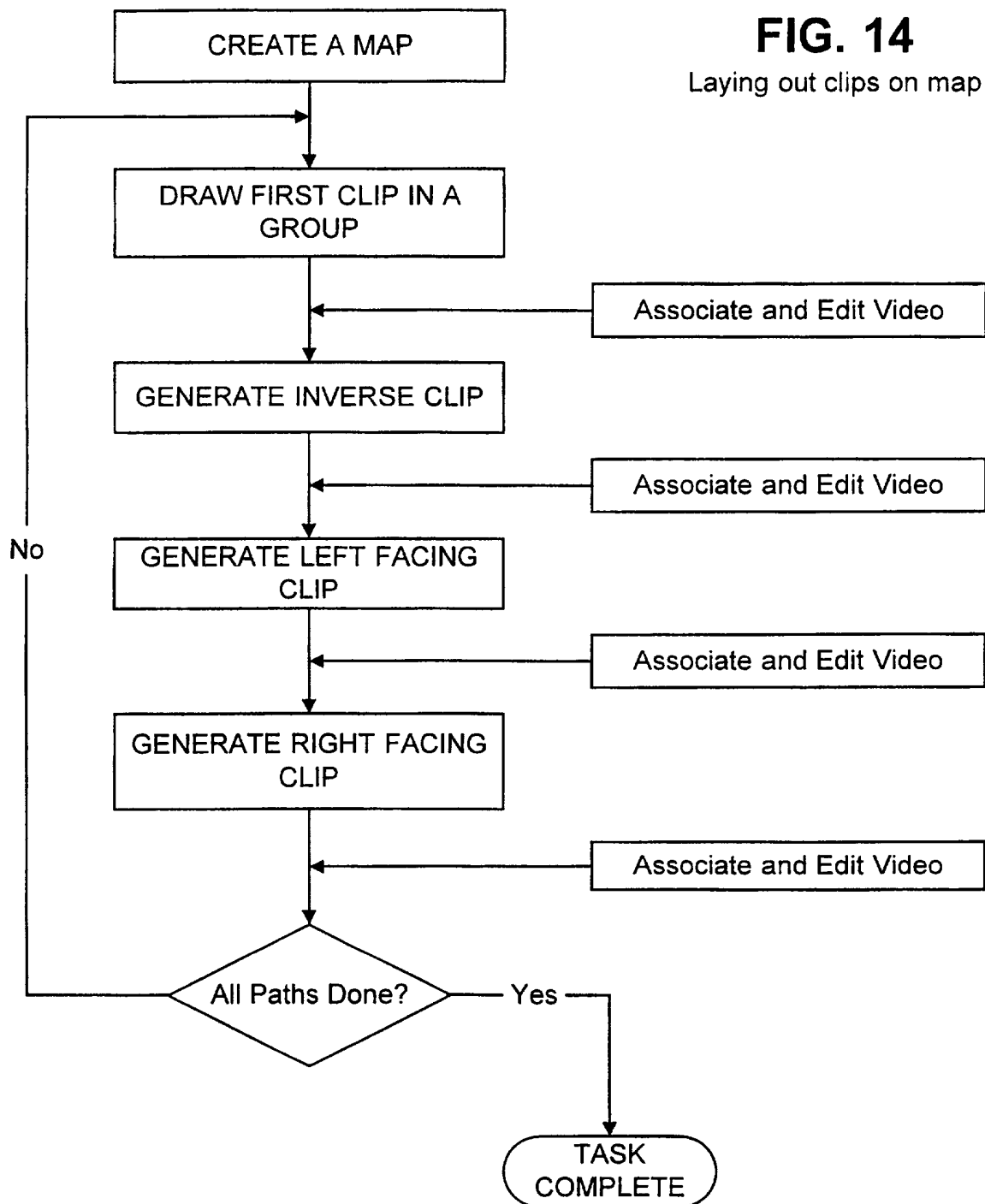
**FIG. 12**  
Spatial Video Edit  
Process



**FIG. 13**  
Spatial Video Map

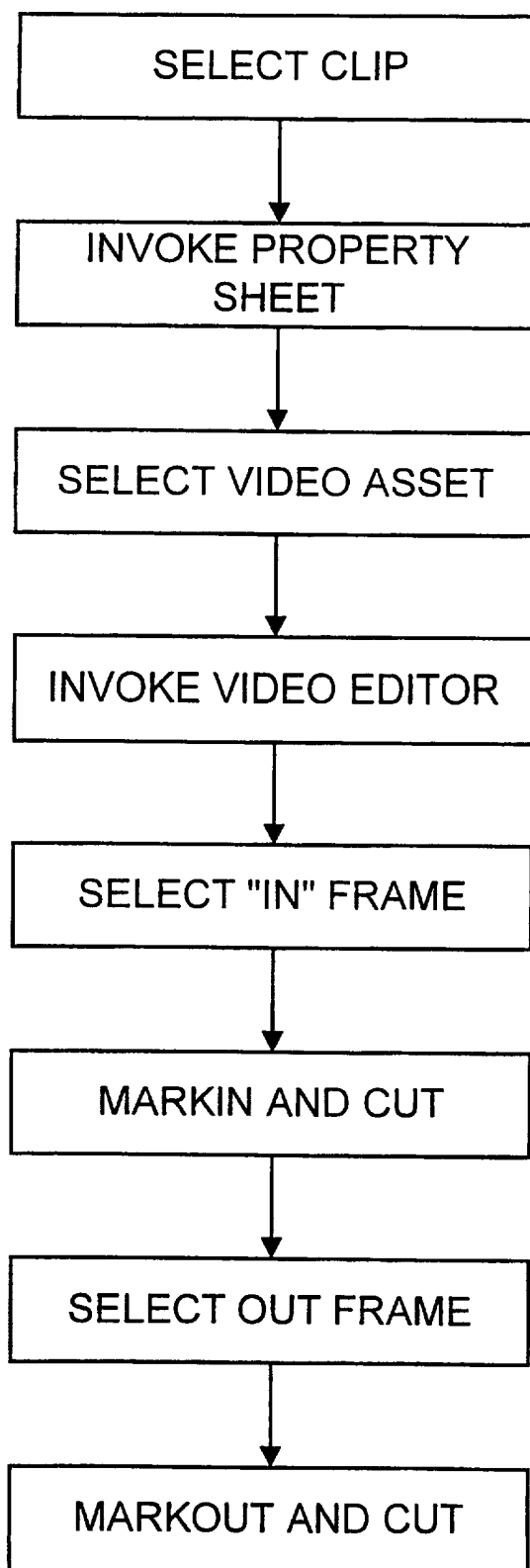
**FIG. 14**

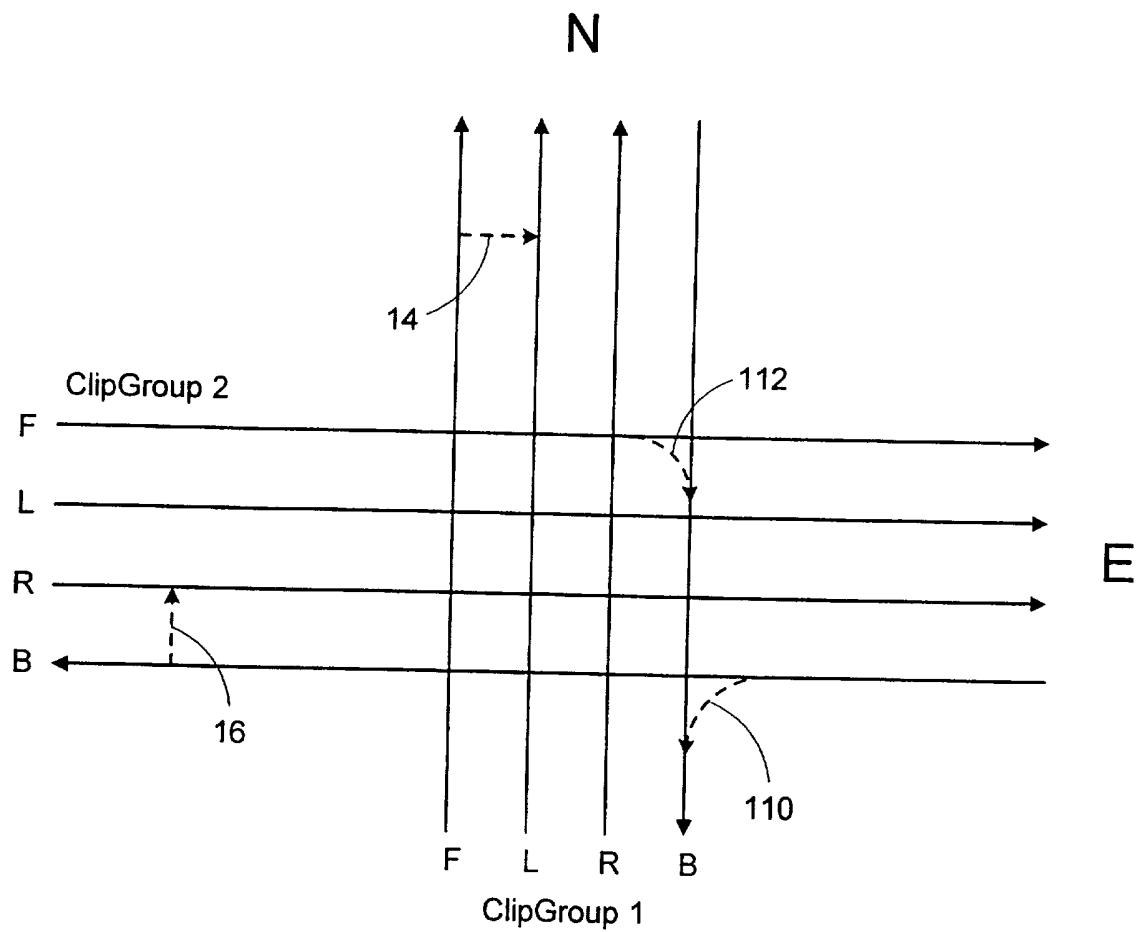
Laying out clips on map



## FIG. 15

Associating videoclip with map

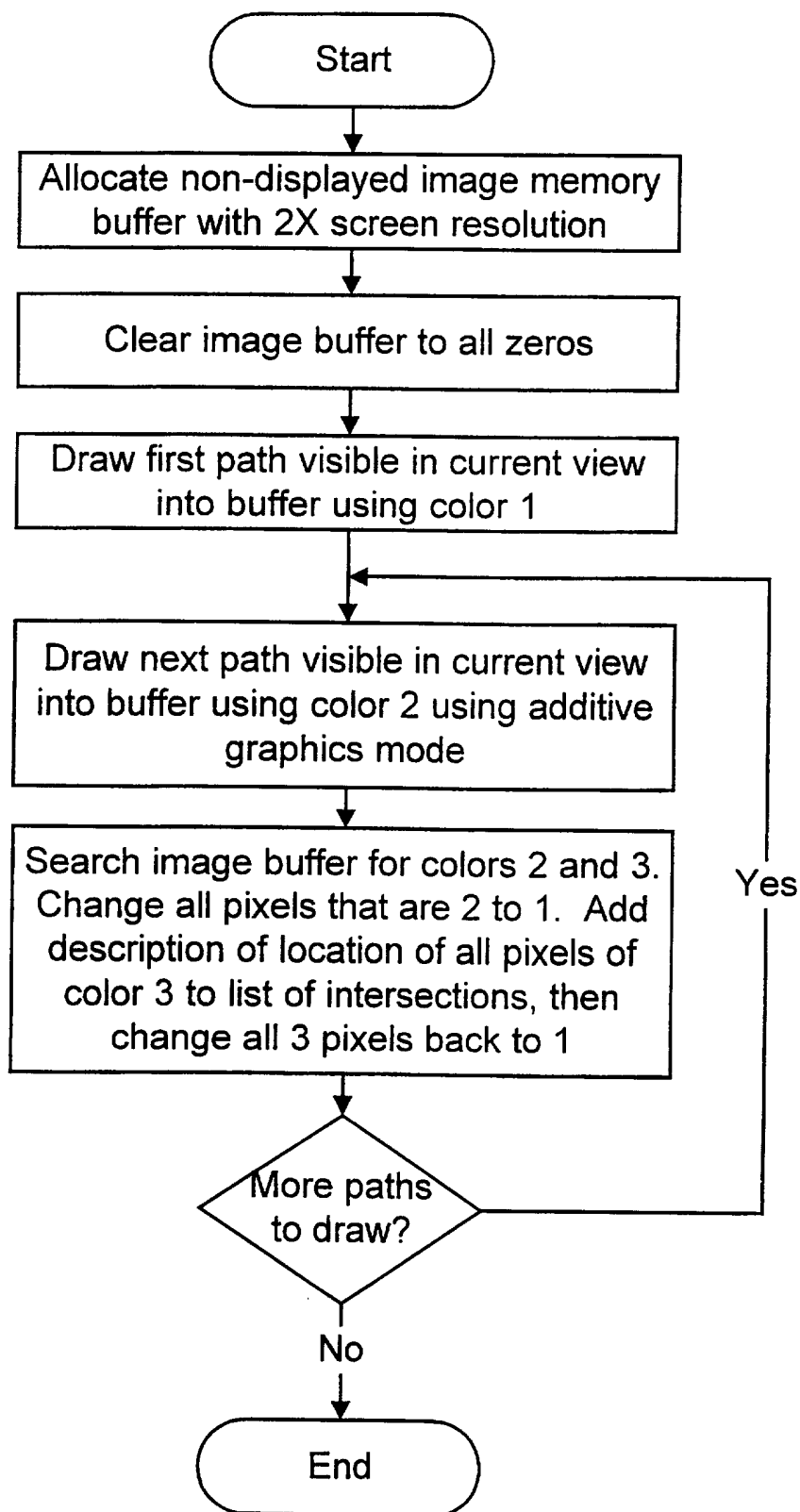




**FIG. 16**  
Intersection of two clip groups

**FIG. 17**

Clip intersection identification





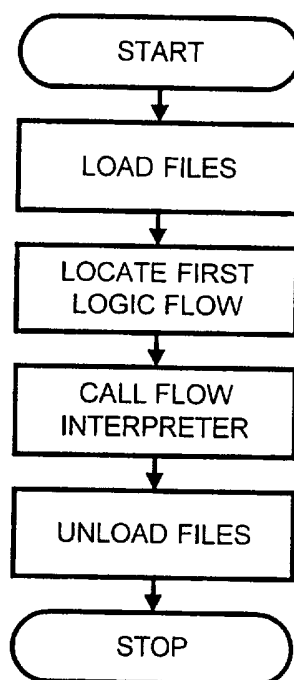


FIG. 18  
Runtime Process

## FIG. 19

Flow (Recursive op code) interpreter

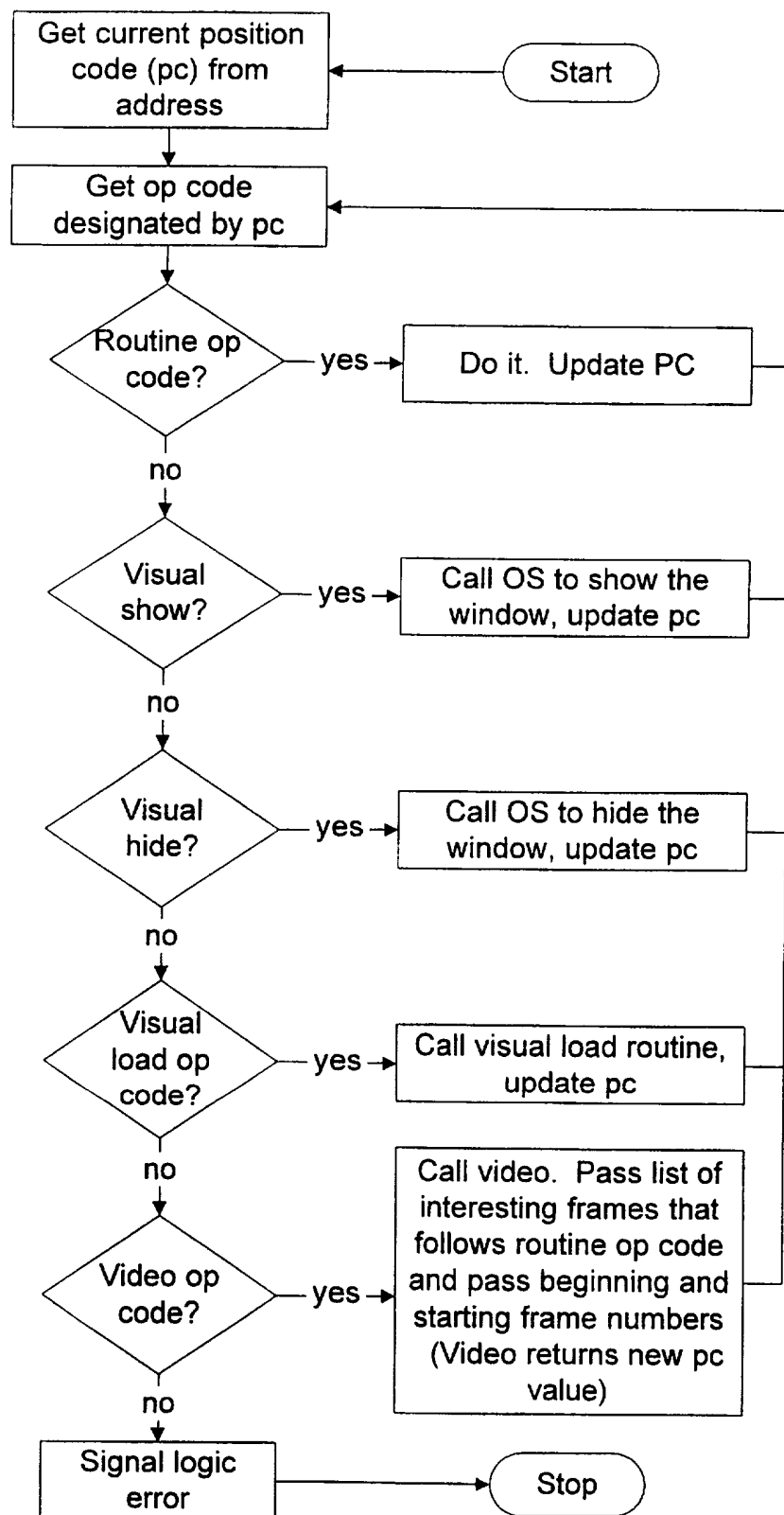
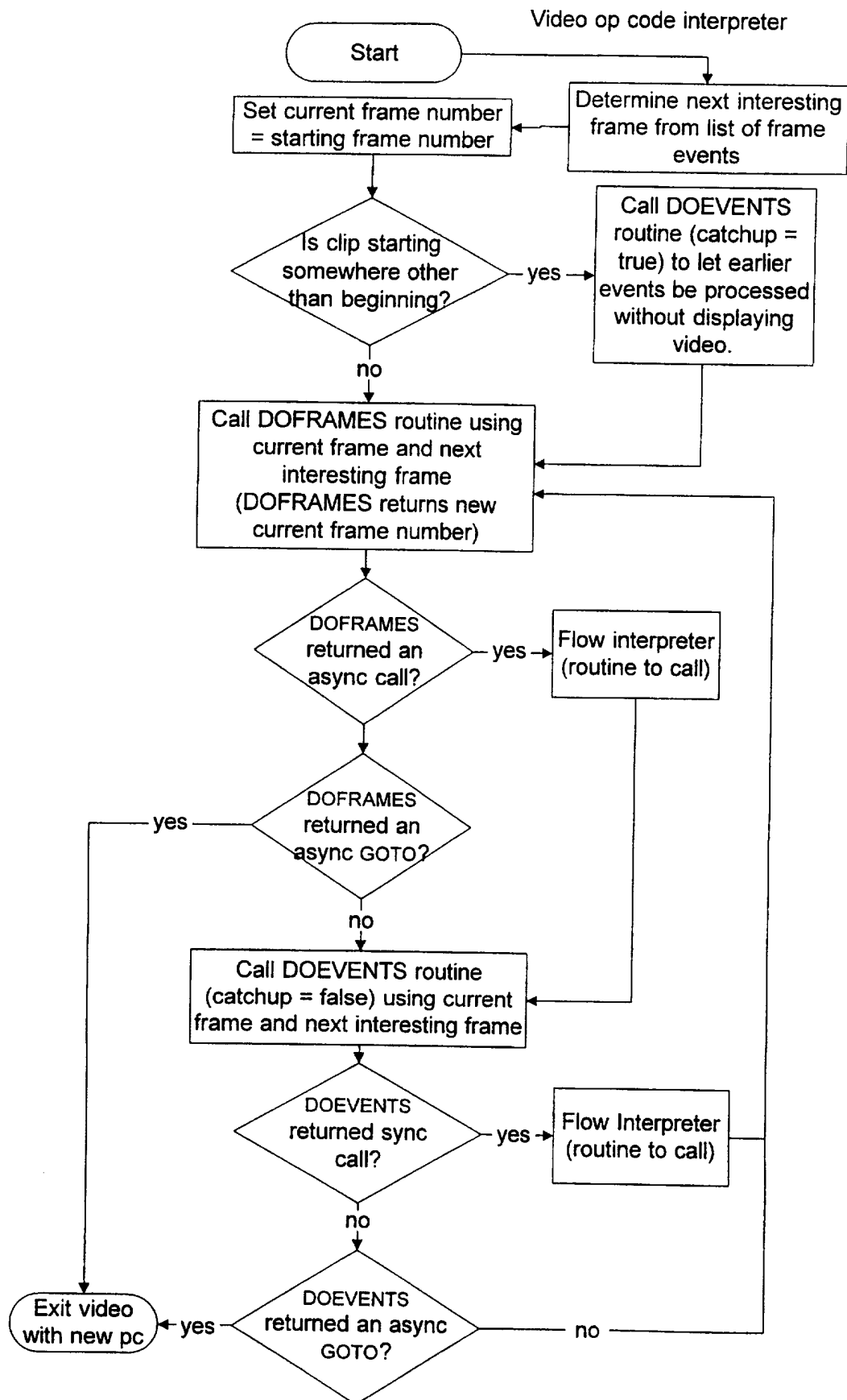
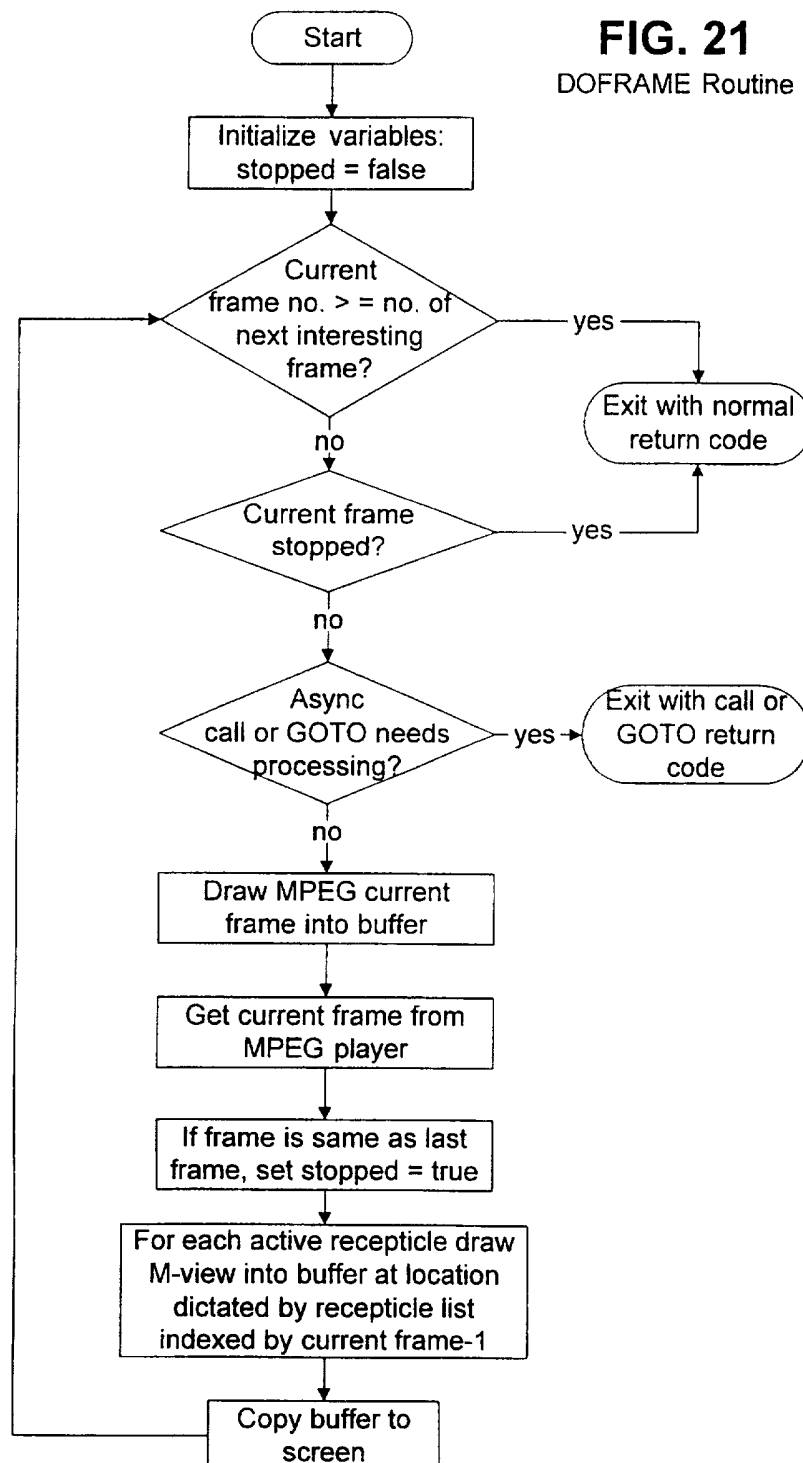


FIG. 20



**FIG. 21**  
DOFRAME Routine



**FIG. 22**  
DOEVENTS routine

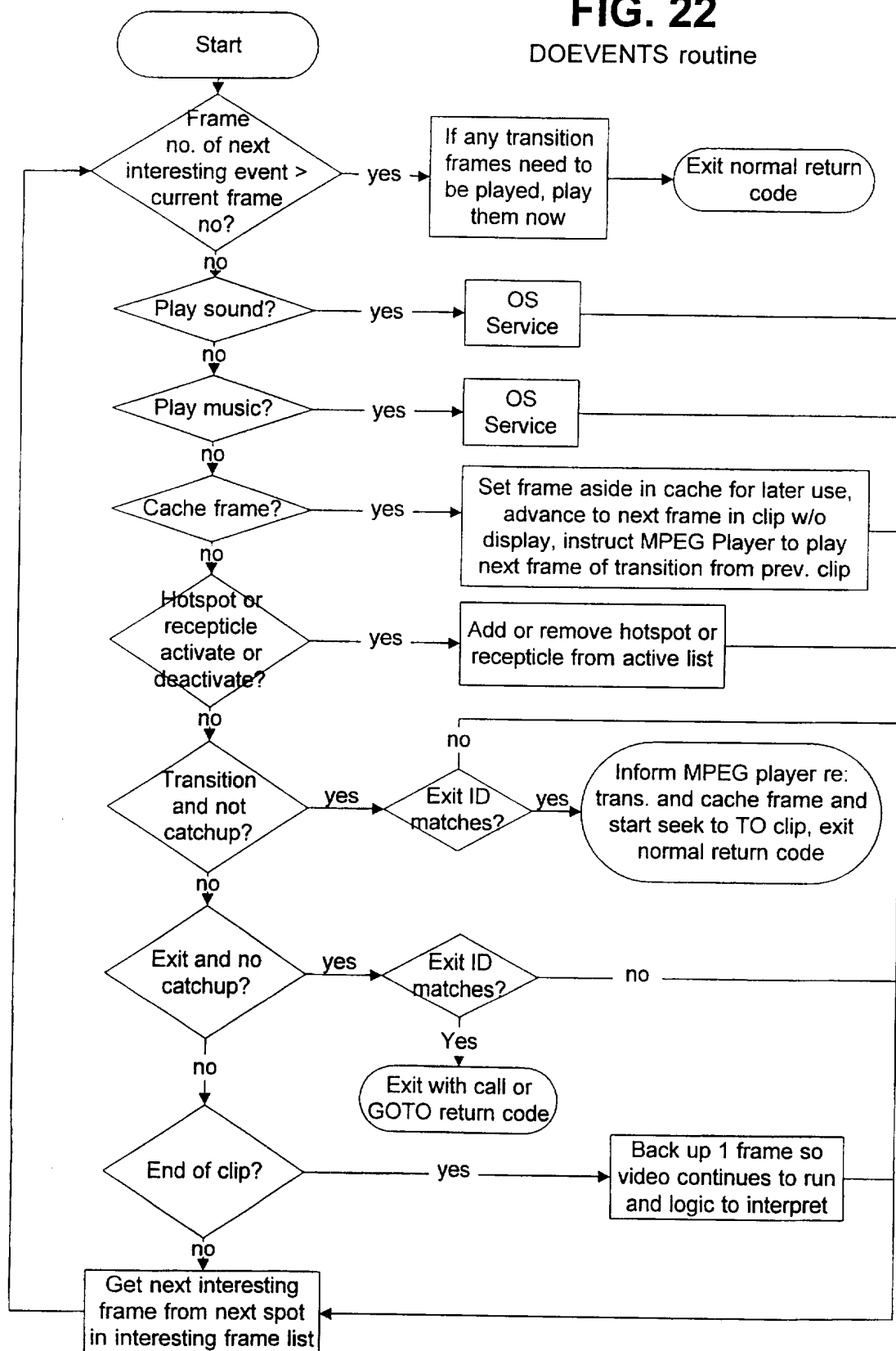
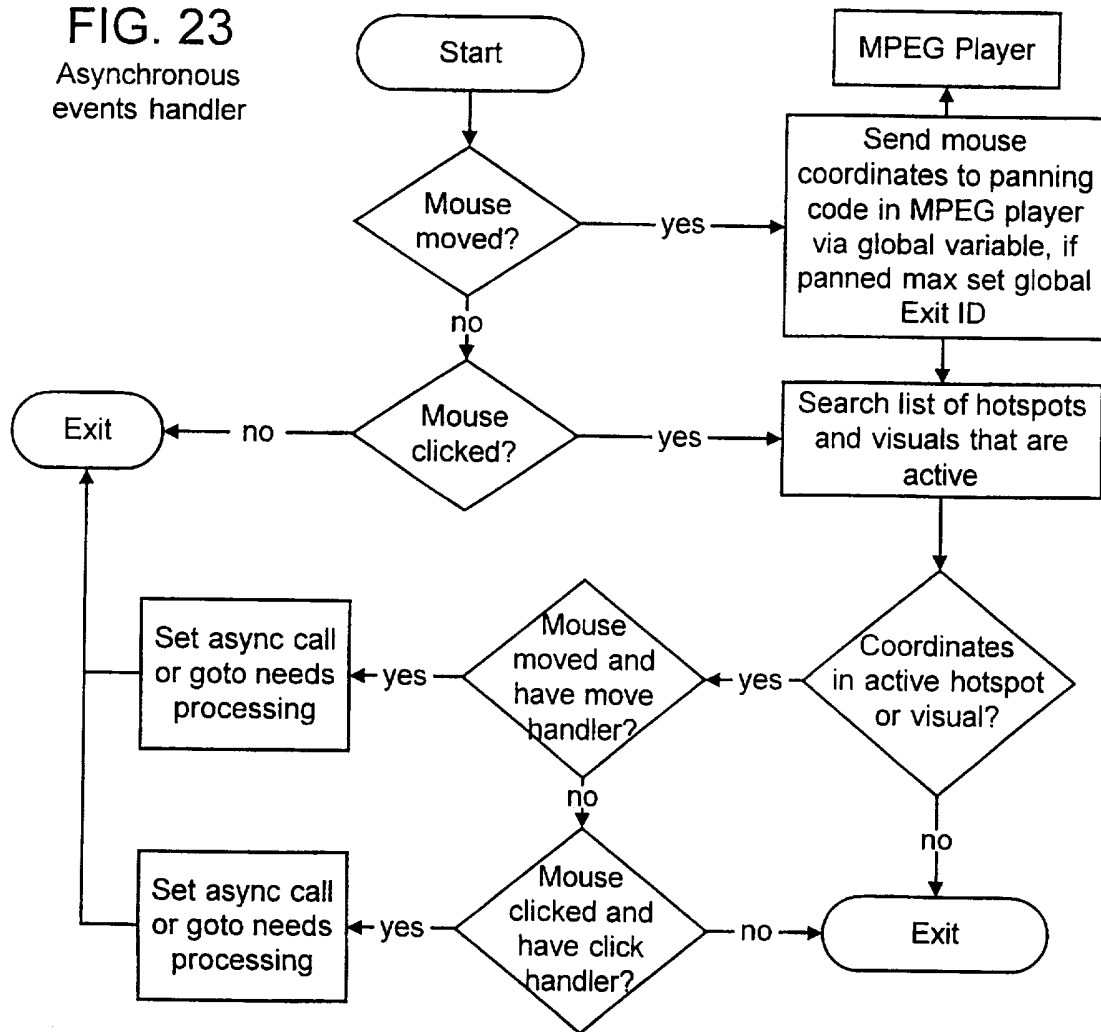
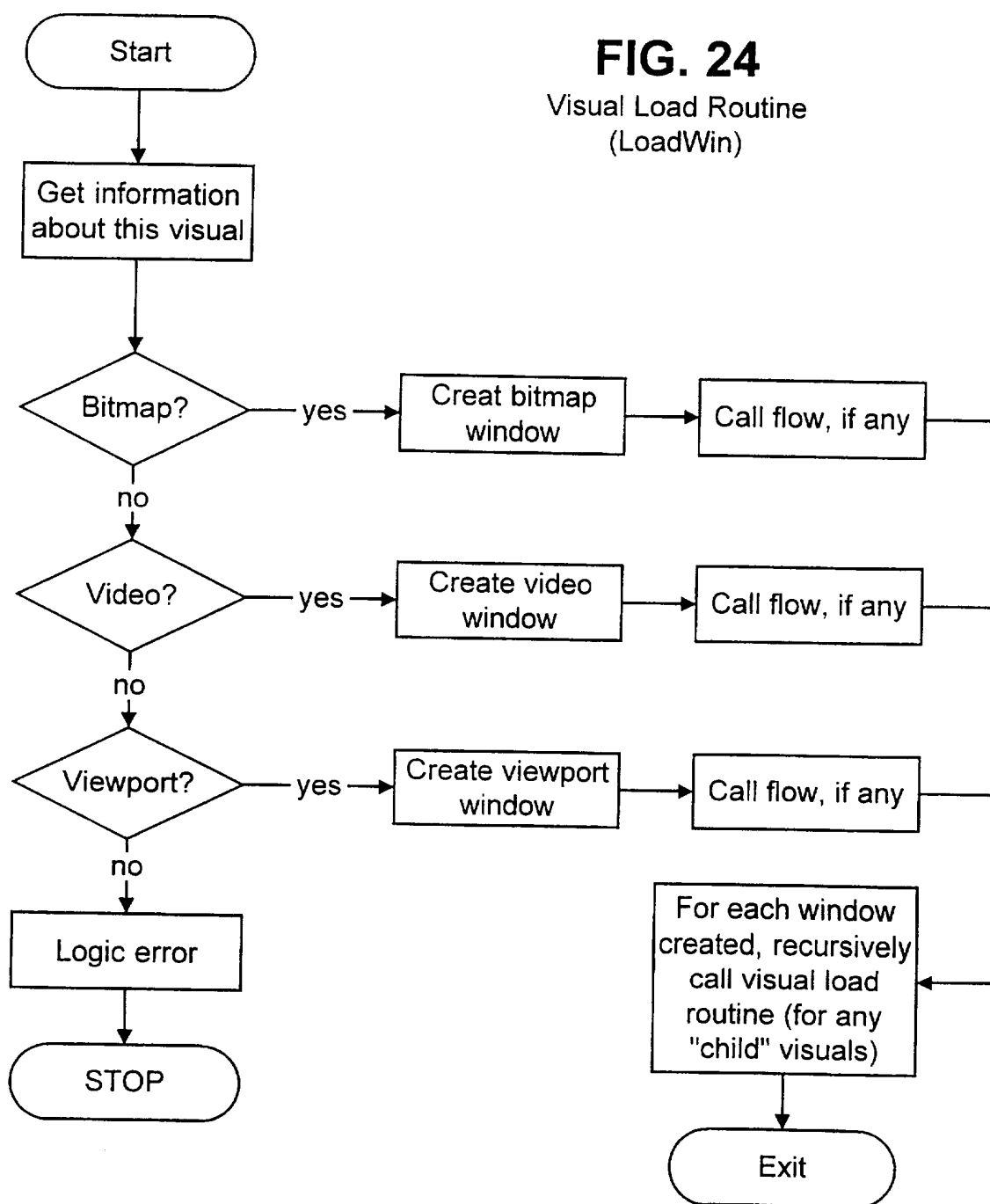


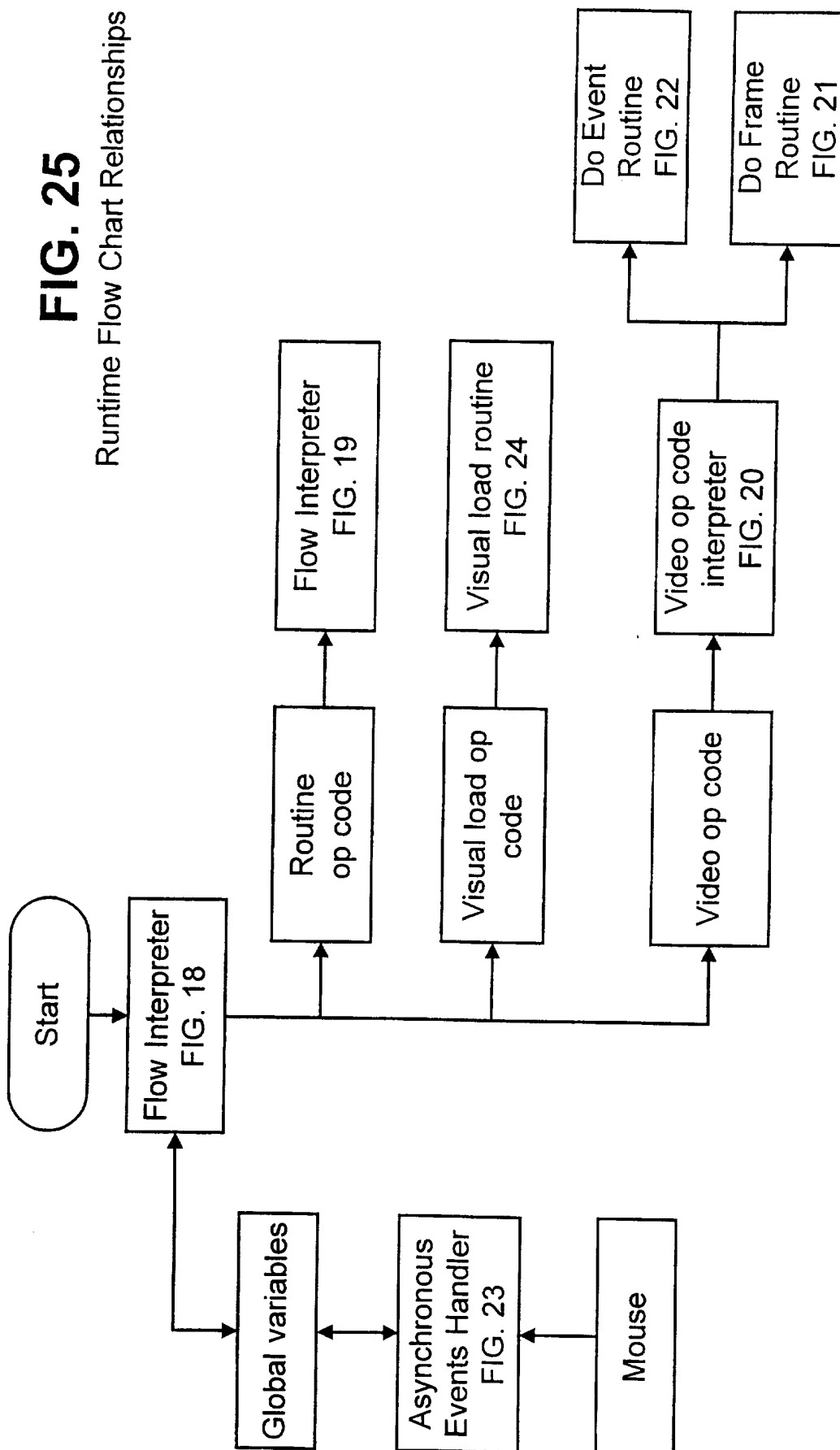
FIG. 23

Asynchronous  
events handler



**FIG. 25**

Runtime Flow Chart Relationships





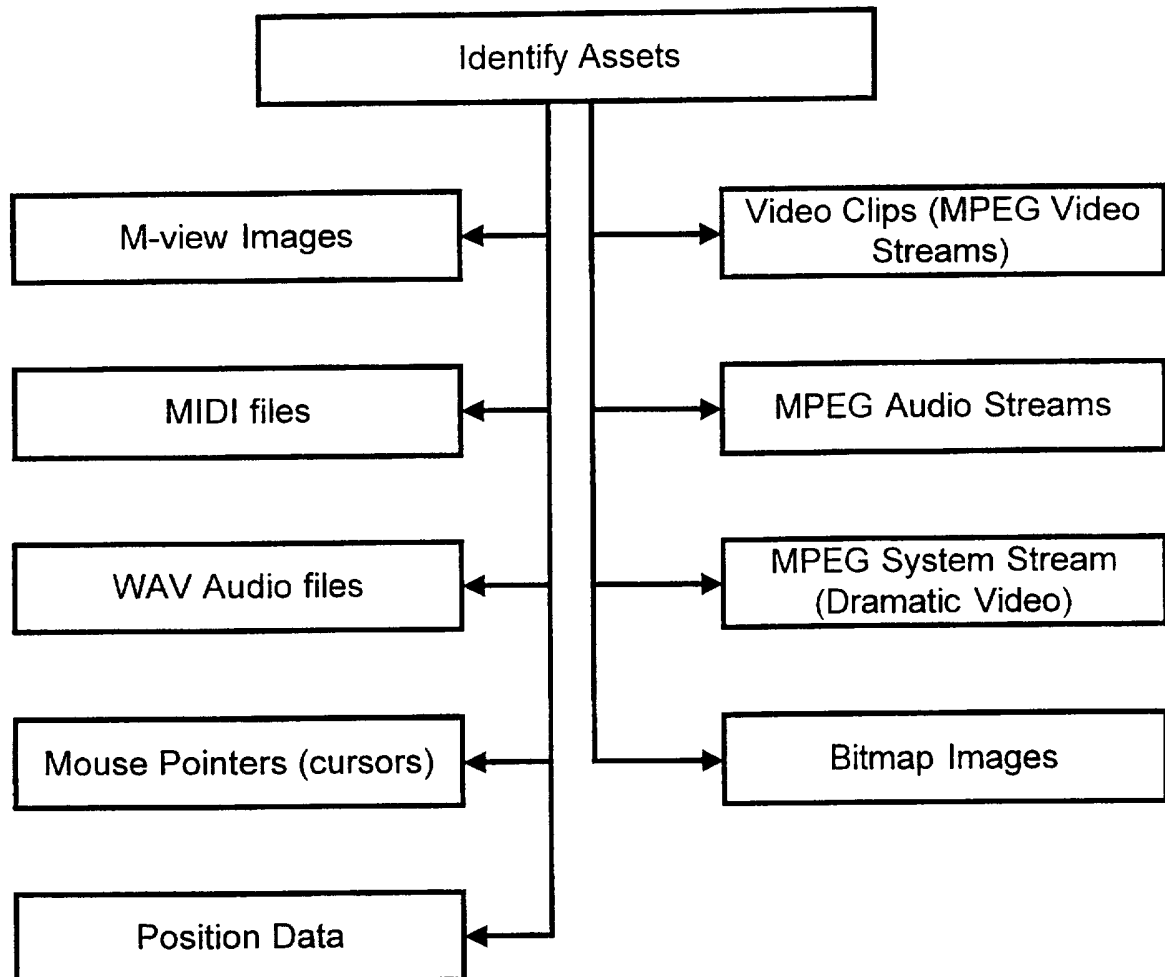
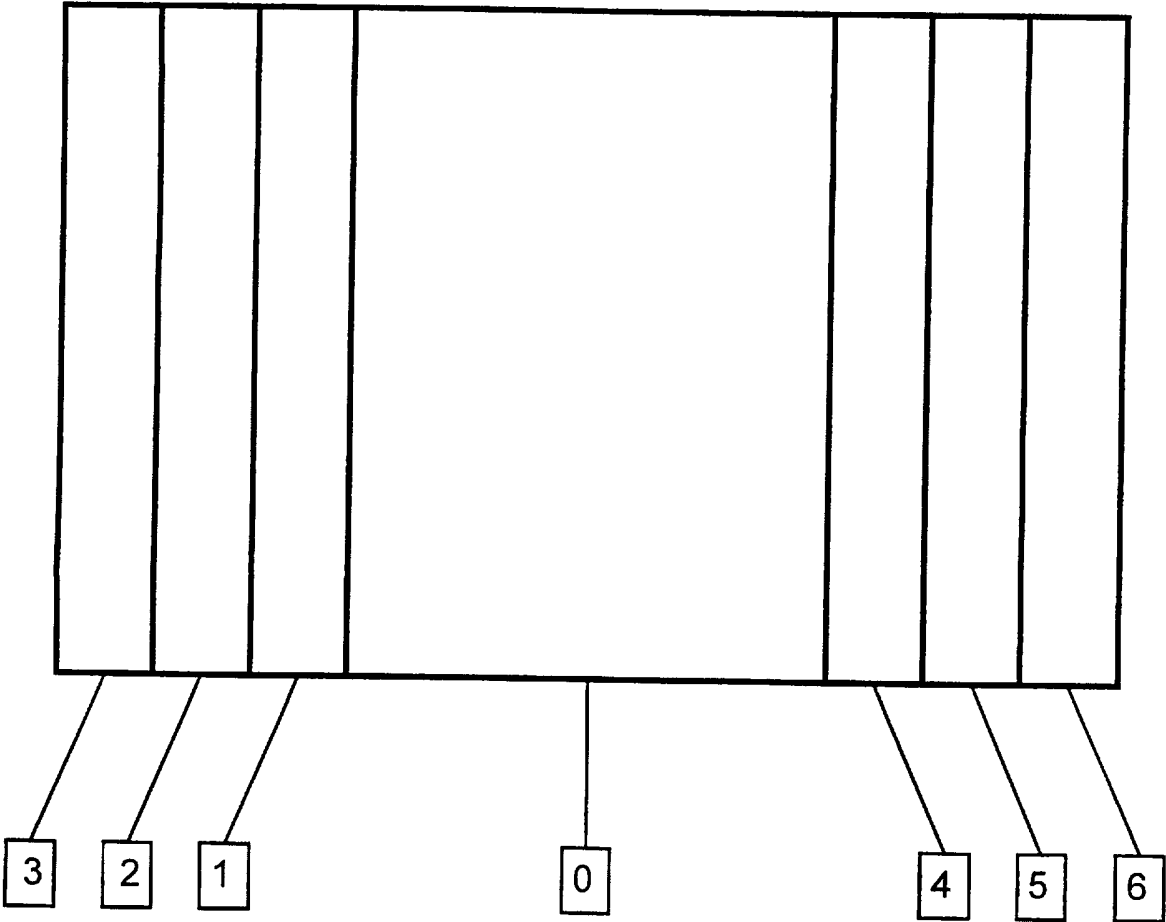


FIG. 26

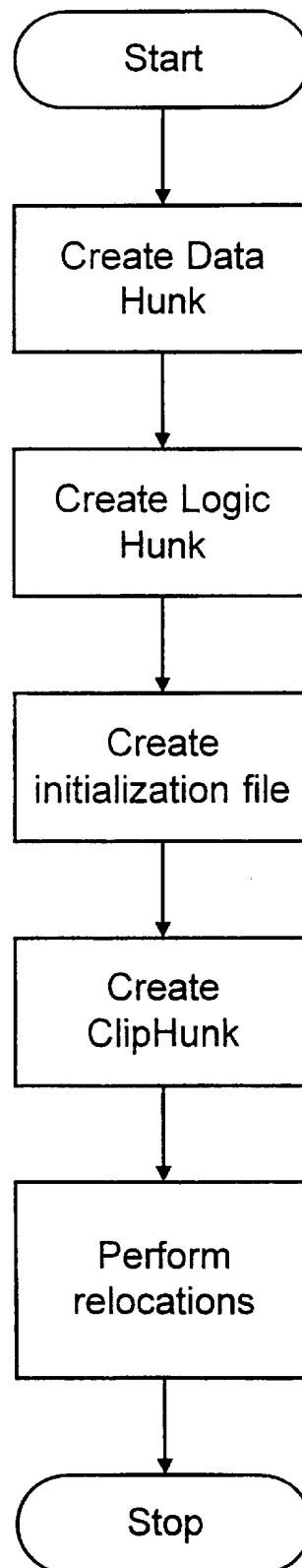
Asset Identification

**FIG. 27**  
Screen Navigation Regions



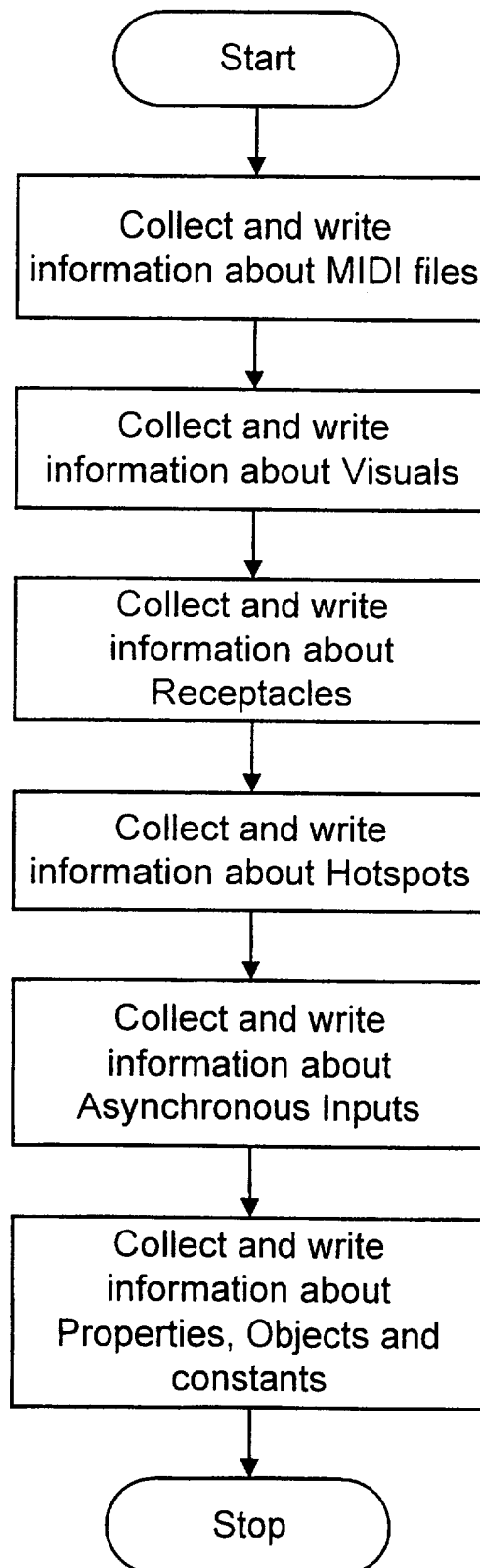
**FIG. 28**

## Binder Process



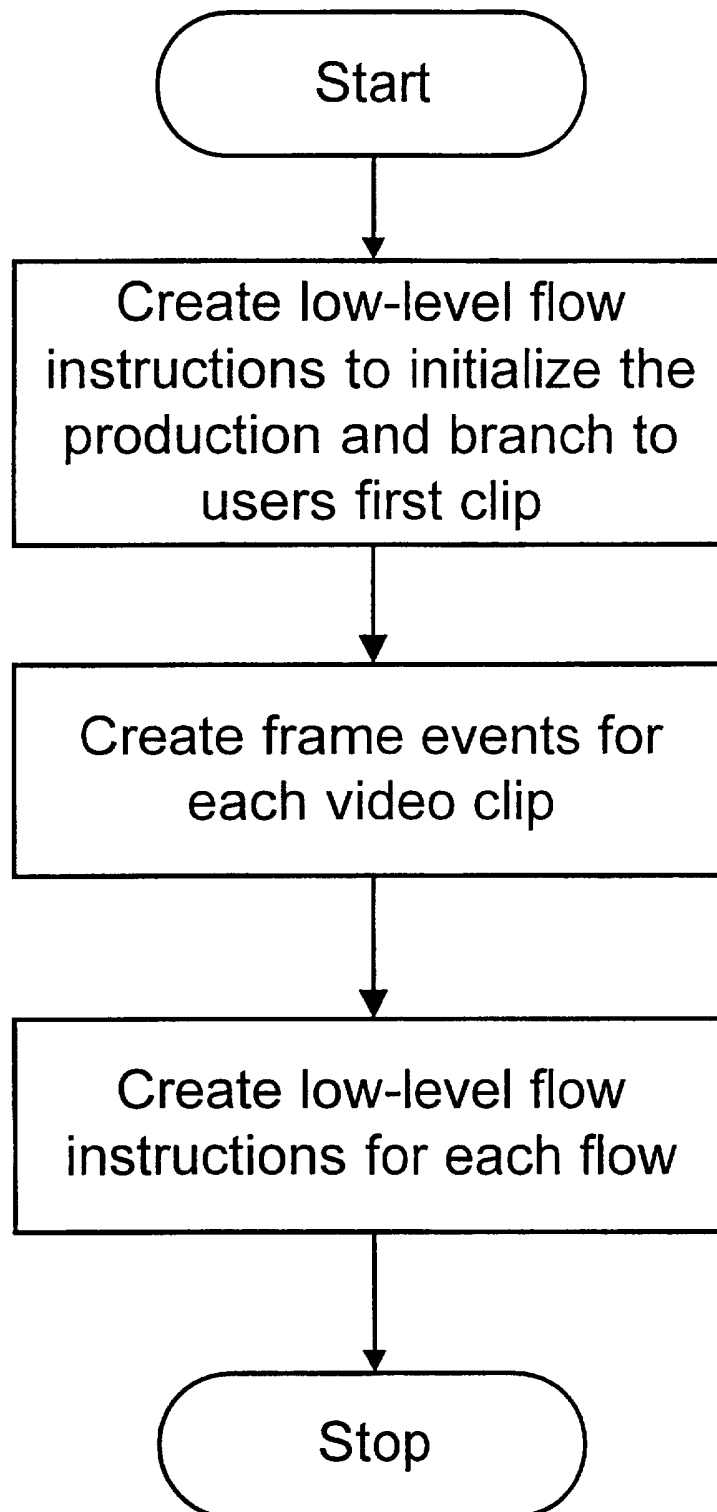
**FIG. 29**

## Data Hunk Creation



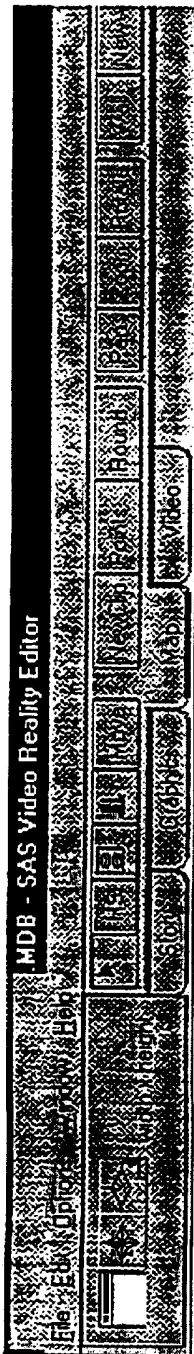
# FIG. 30

## Logic Hunk Creation Process

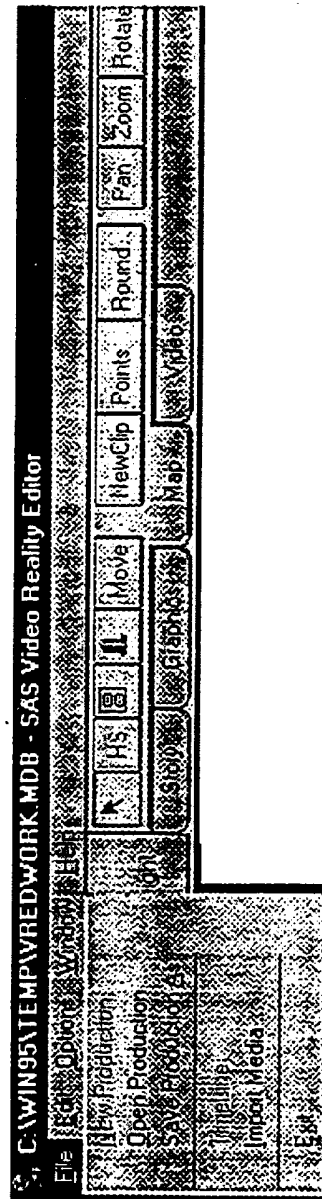


**ATTACHMENT 1**

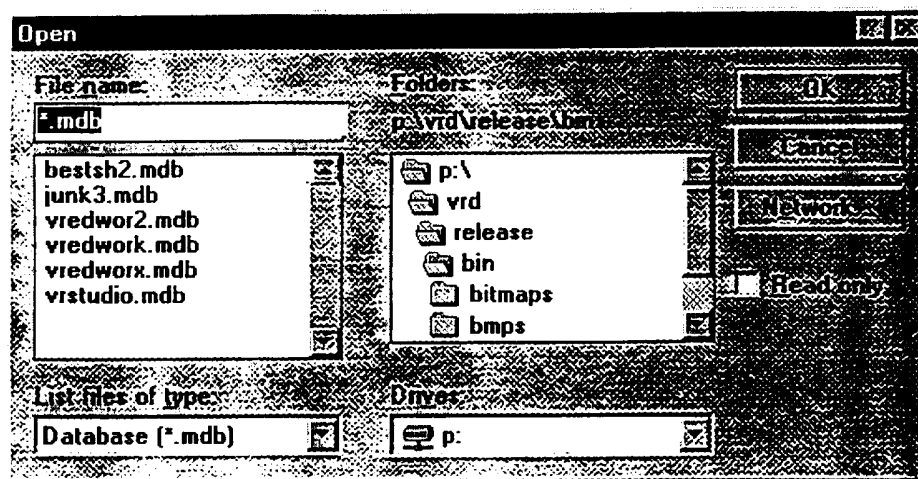
**SPATIAL VIDEO EDIT SYSTEM  
GRAPHICAL USER INTERFACE**



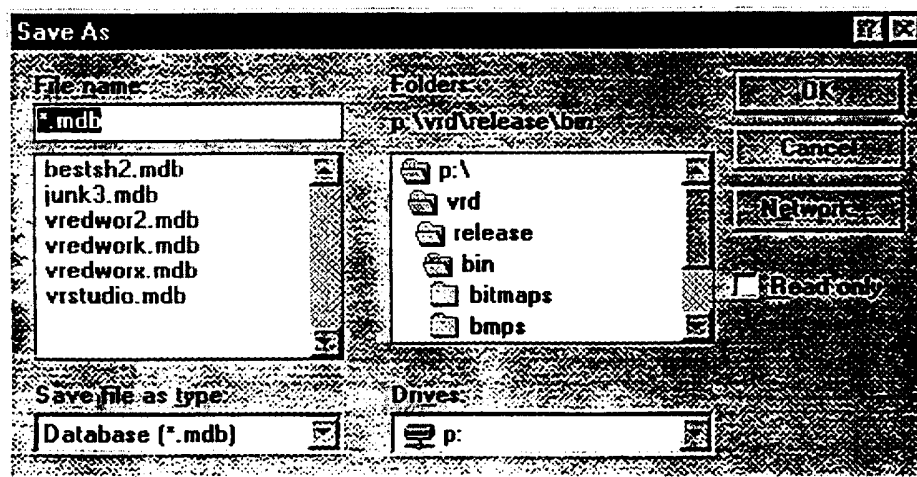
Video Reality Editor Primary Screen



Primary Screen File

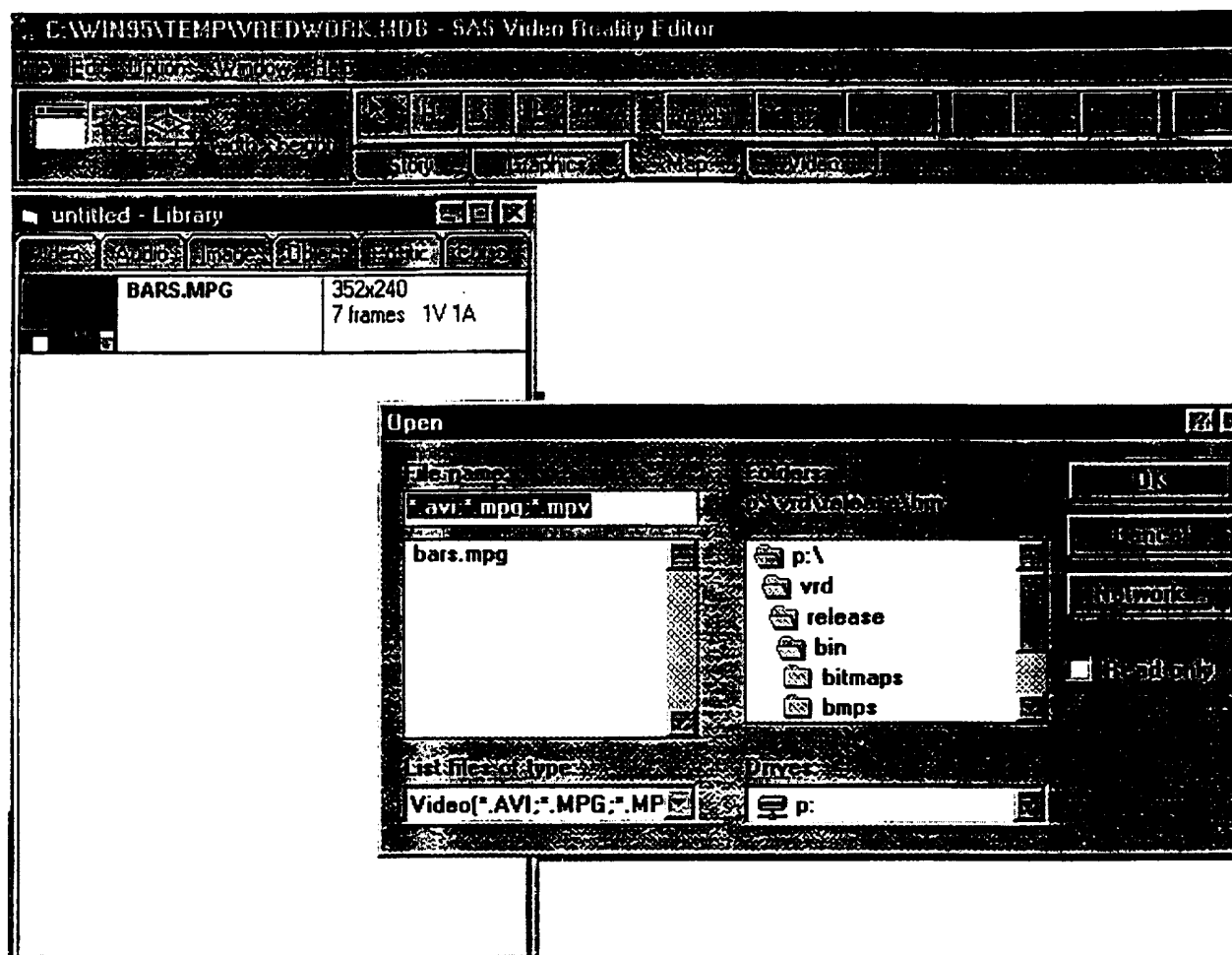


Primary Screen File Open Production

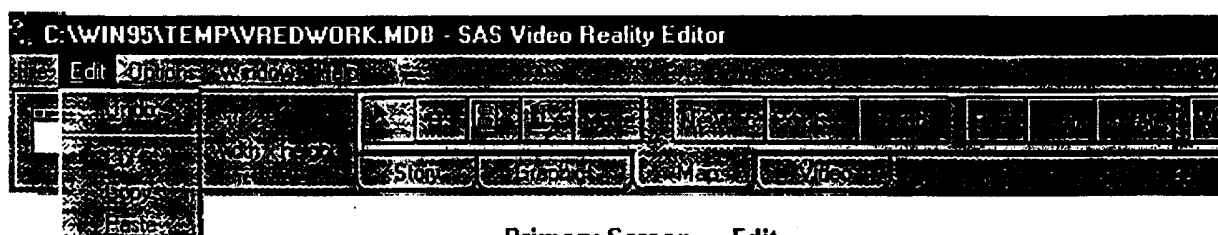


Primary Screen File Save Production As...

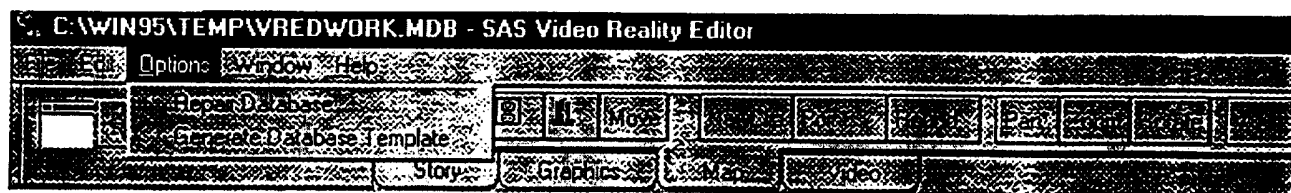




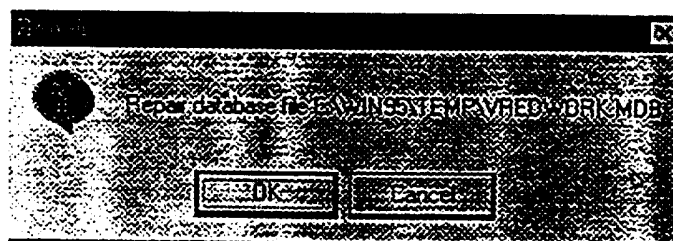
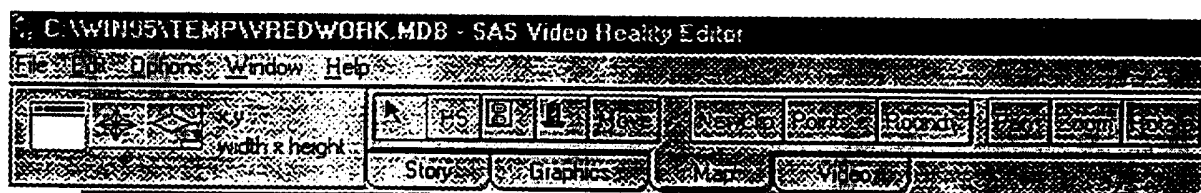
Primary Screen File Import Media



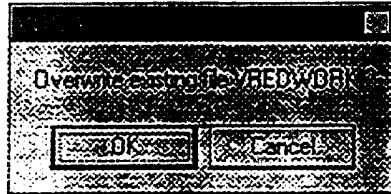
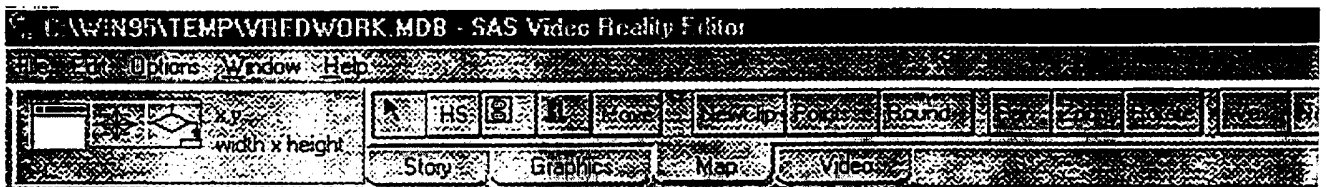
Primary Screen Edit



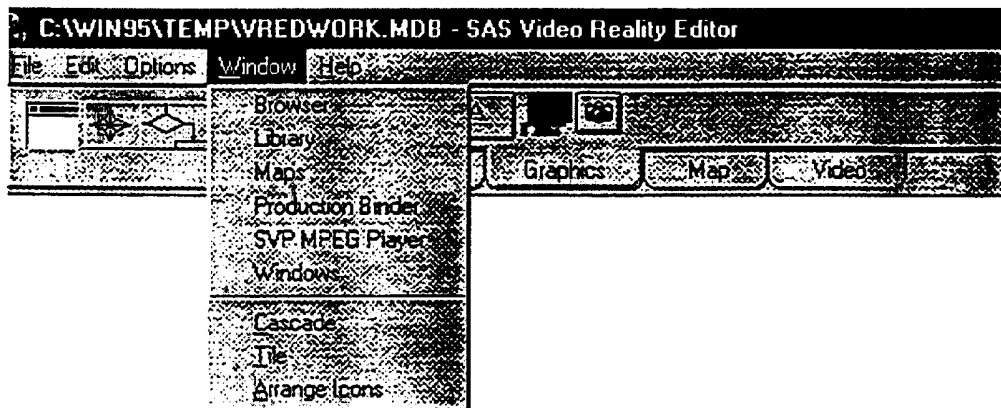
Primary Screen Options



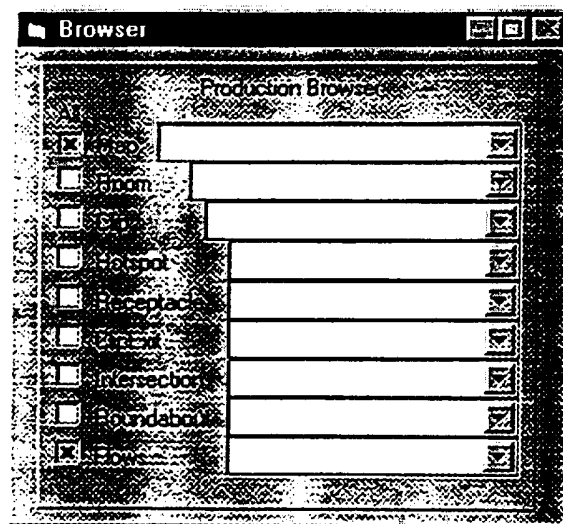
Primary Screen Options Repair Database



Primary Screen   Options   Generate Database Template



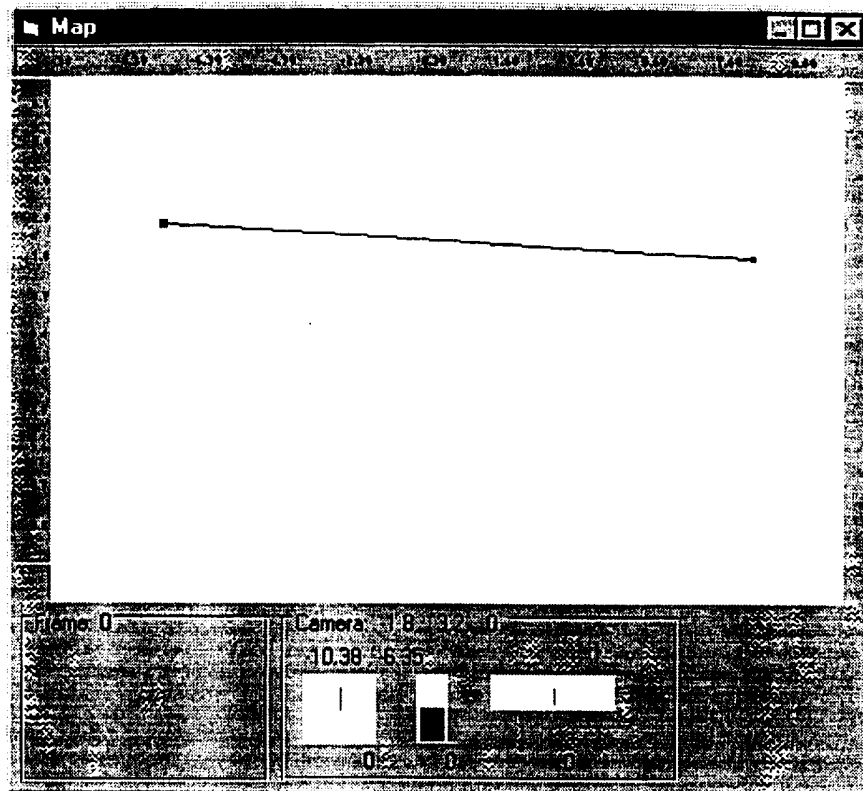
Primary Screen   Window



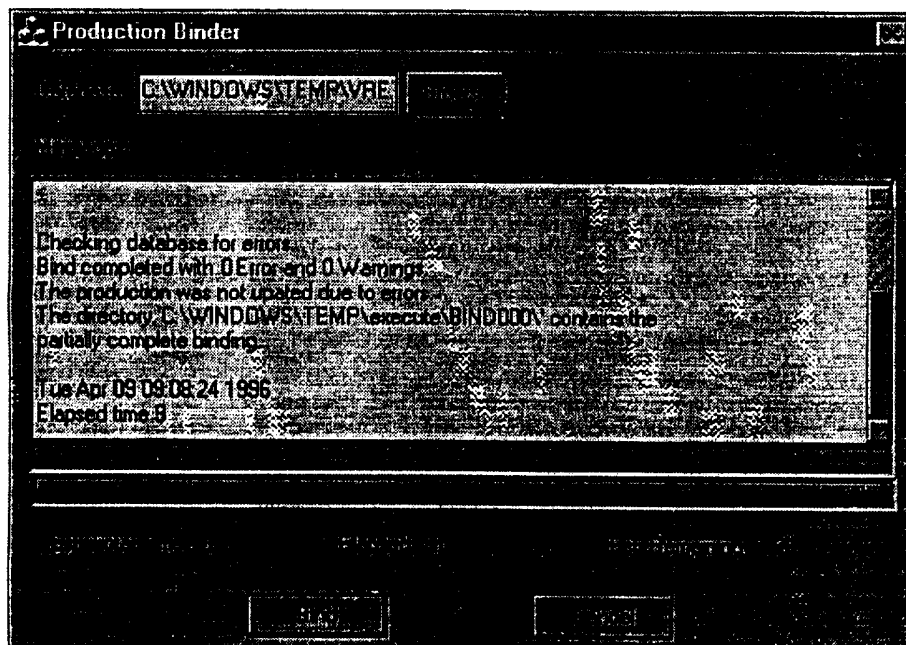
Primary Screen Window Browser

untitled - Library		
Video	Audio	Image
	MVCAM2FW.MPG	352x240 39 frames 1V 1A
	MVCAM2LF.MPG	352x240 39 frames 1V 1A
	MVCAM2RG.MPG	352x240 39 frames 1V 1A
	MVCAMBA1.MPG	352x240 49 frames 1V 1A
	MVCAMBAC.MPG	352x240 49 frames 1V 1A
	MVCAMFWD.MPG	352x240 49 frames 1V 1A
	MVCAMLFT.MPG	352x240 49 frames 1V 1A
	MVCAMRGH.MPG	352x240 49 frames 1V 1A
	RTTURNBA.MPG	352x240 46 frames 1V 1A

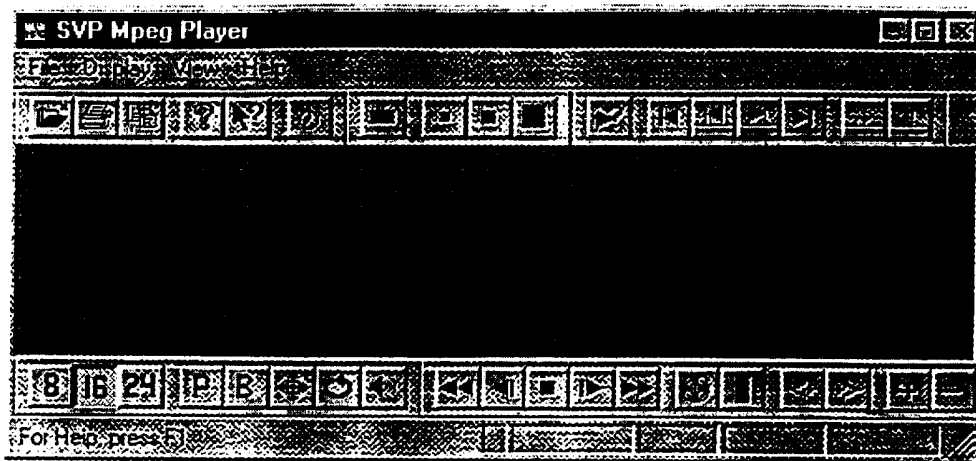
Primary Screen Window Library



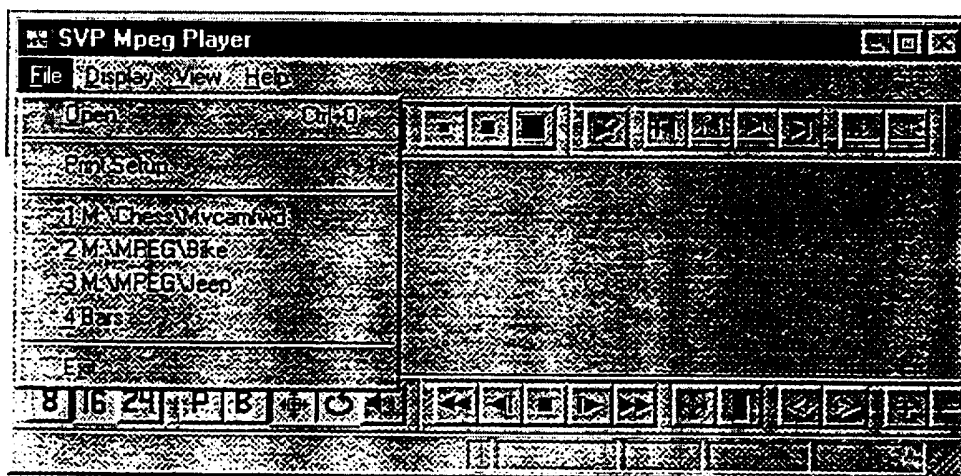
Primary Screen Window Maps



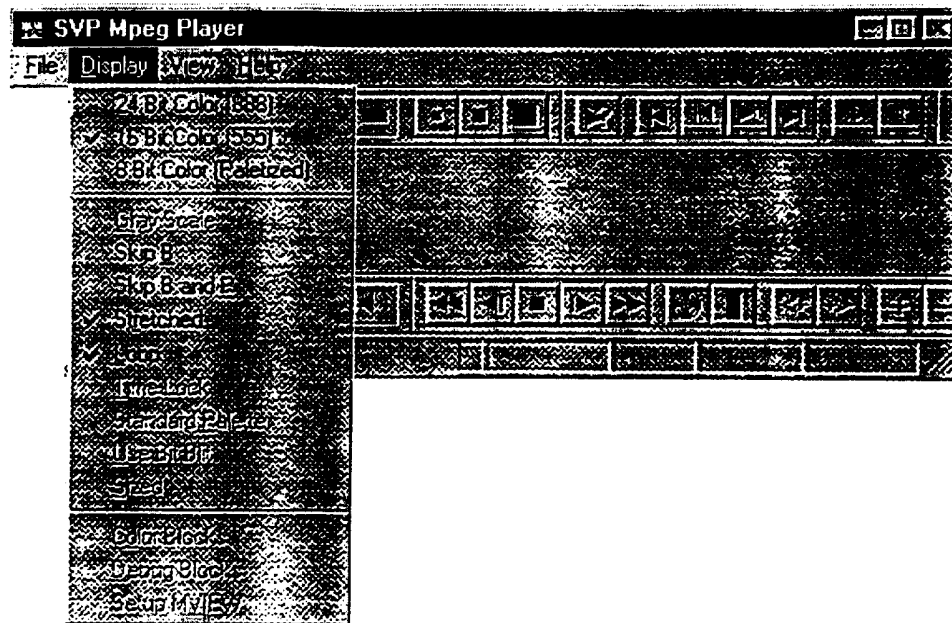
Primary Screen Window Production Binder



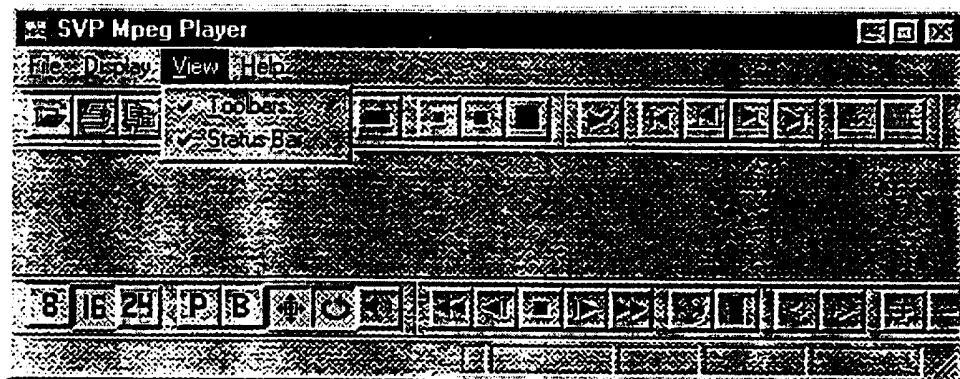
Primary Screen Window SVP Mpeg Player



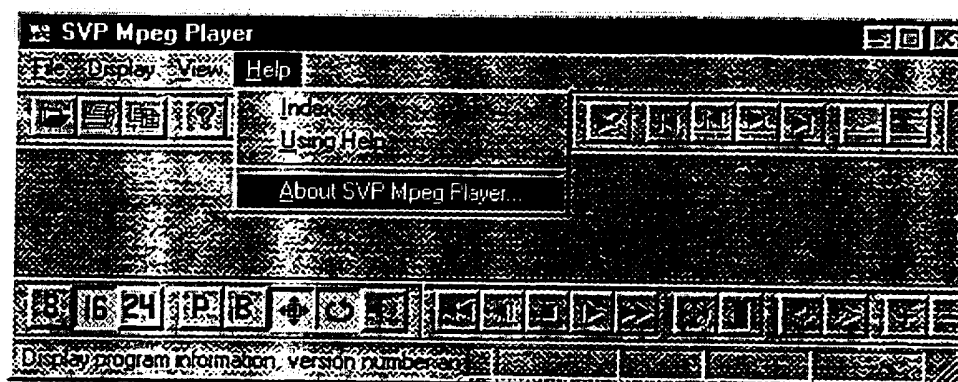
Primary Screen Window SVP Mpeg Player File



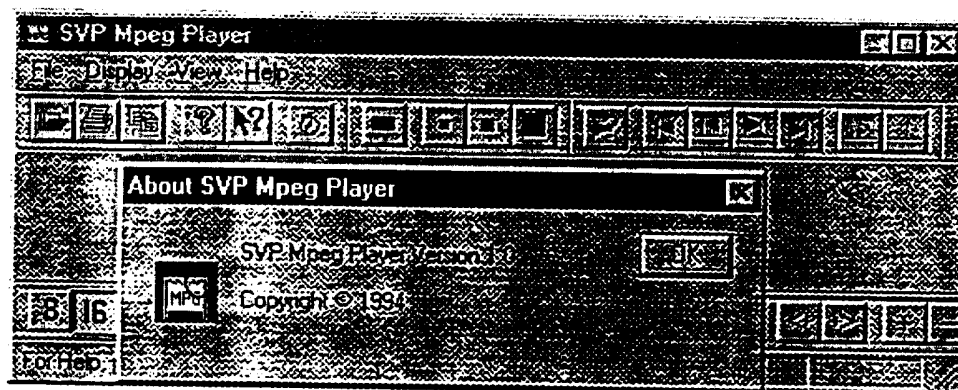
Primary Screen Window SVP Mpeg Player Display



Primary Screen Window SVP Mpeg Player View



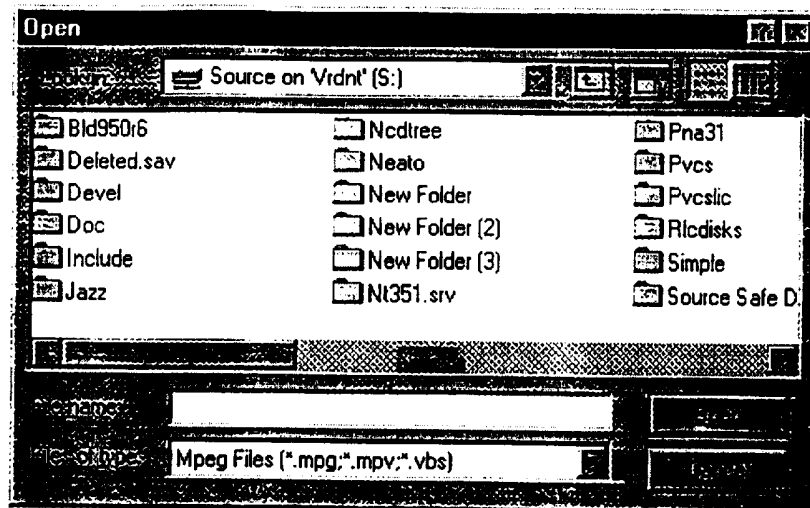
Primary Screen   Window   SVP Mpeg Player   About SVP Mpeg Player



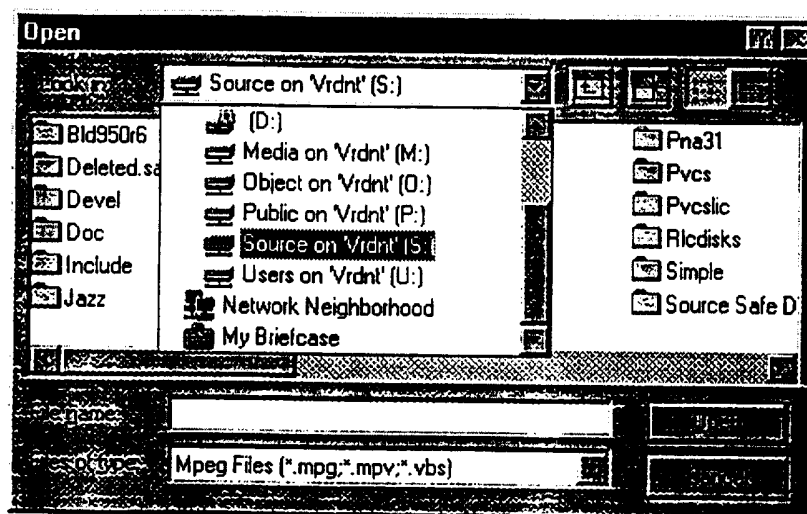
Primary Screen   Window   SVP Mpeg Player   About SVP Mpeg Player



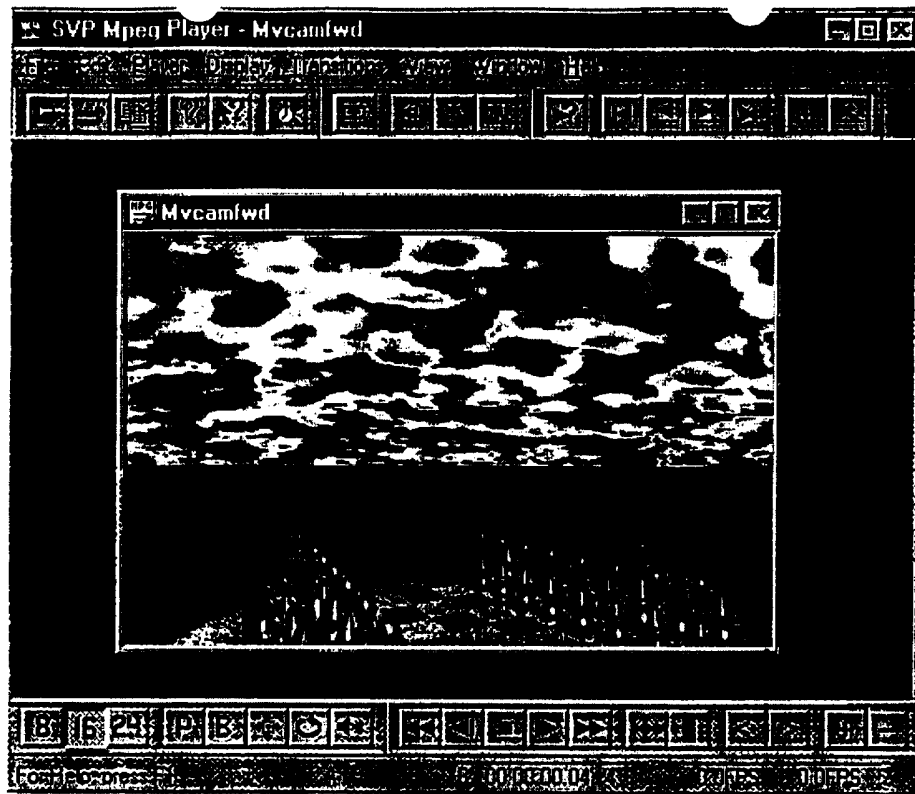
FIGURE P 21



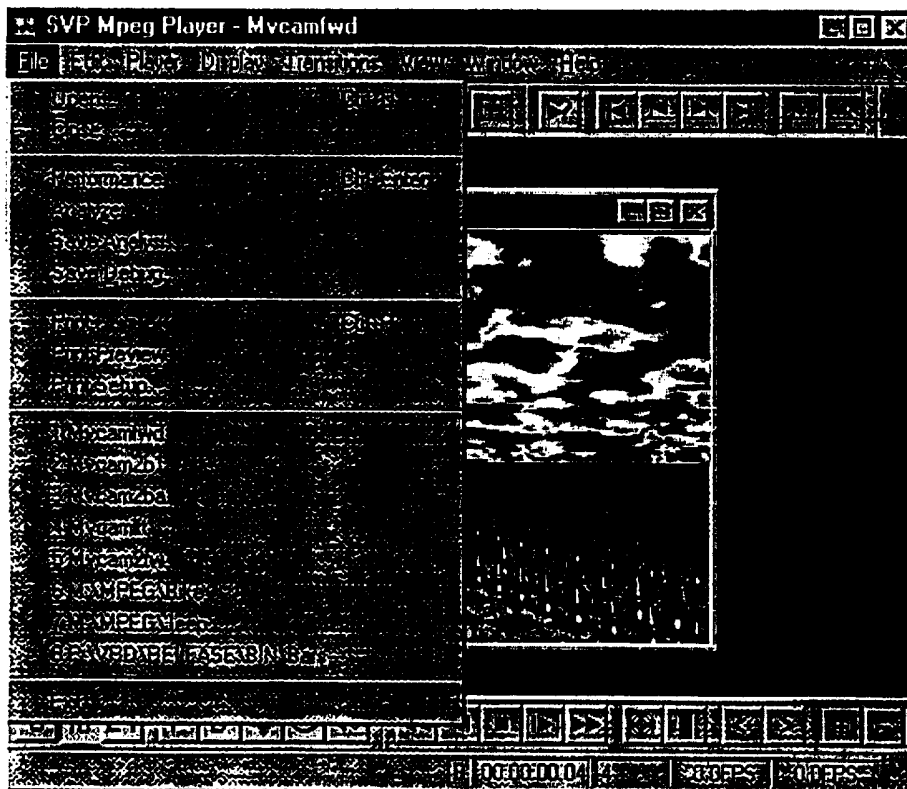
Primary Window SVP Mpeg Player File Open



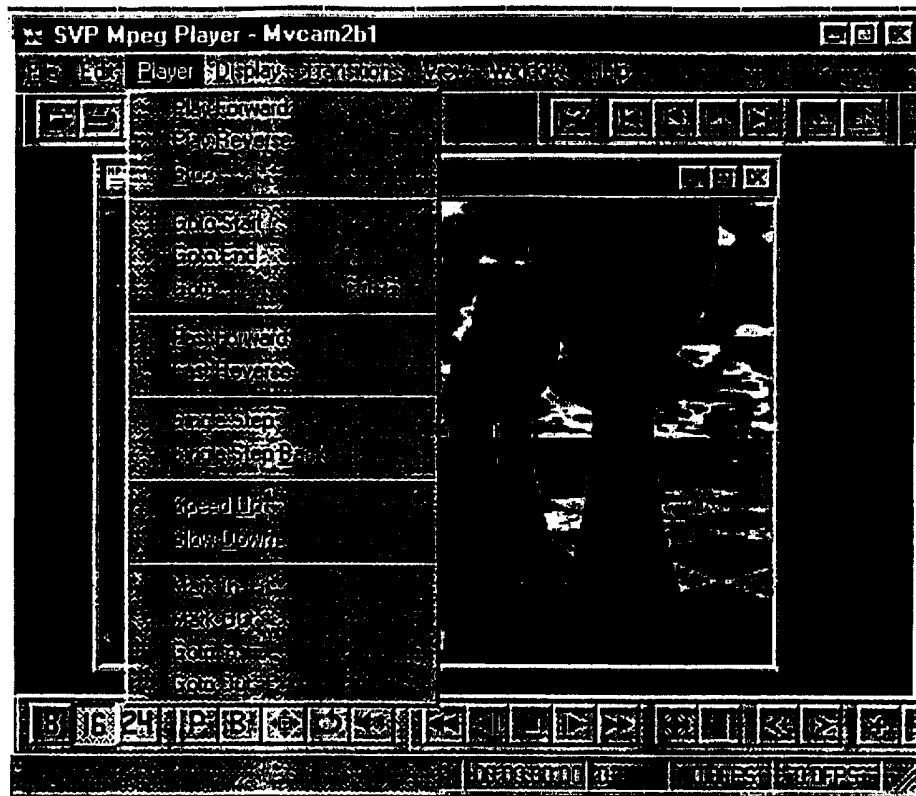
Primary Window SVP Mpeg Player File Open Down Arrow



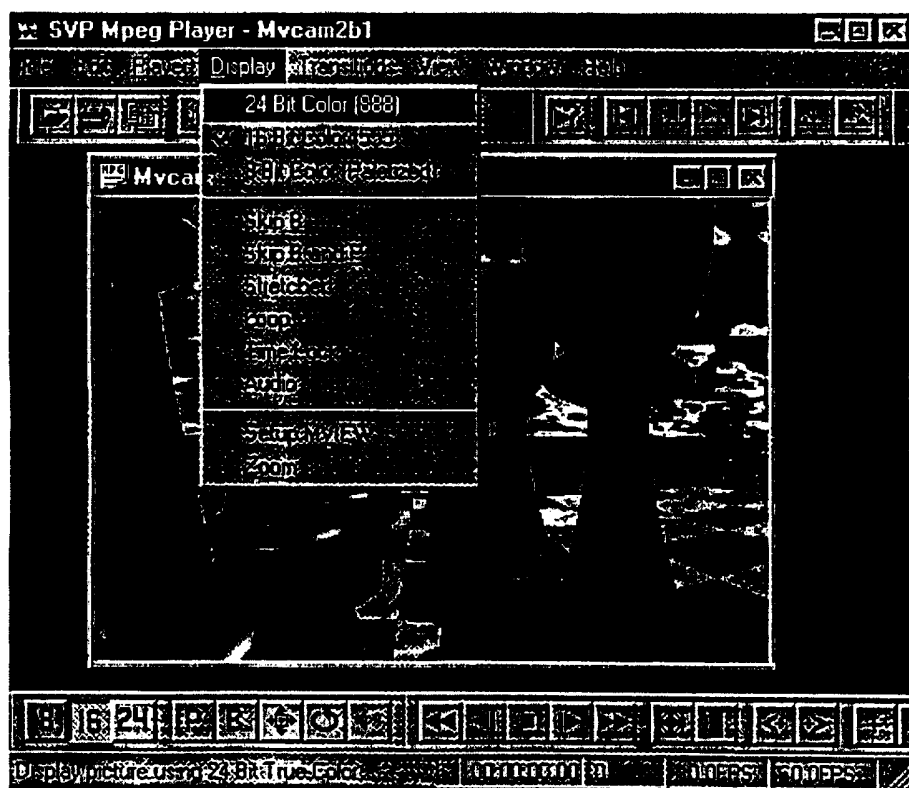
Primary Screen Window SVP Mpeg Player File Open  
Down Arrow (select a file)



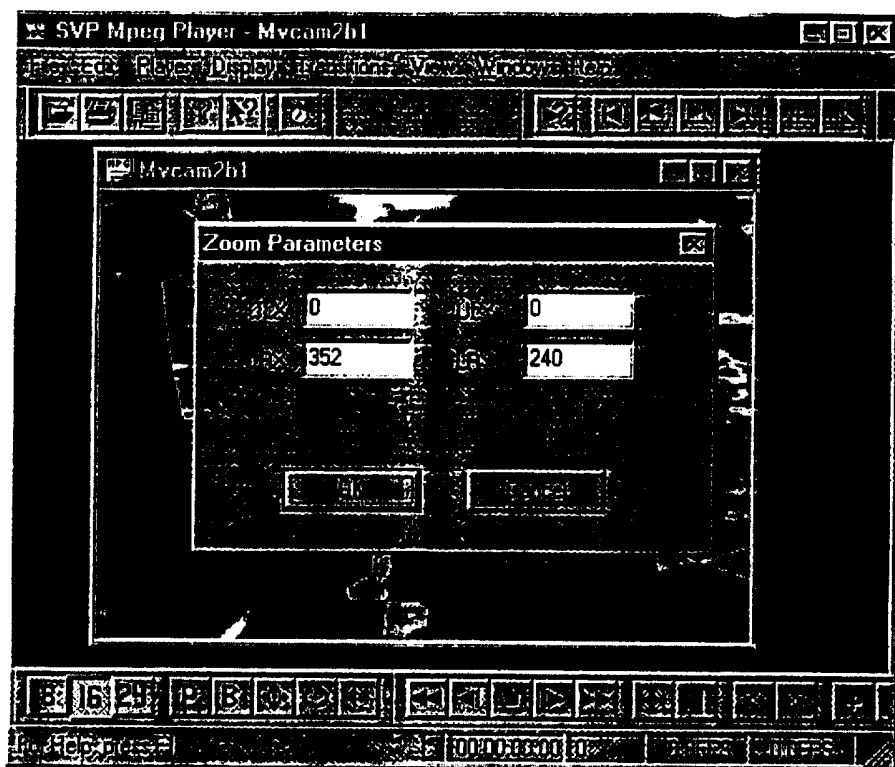
Primary Screen Window SVP Mpeg Player File Open  
Down Arrow (select a file) File



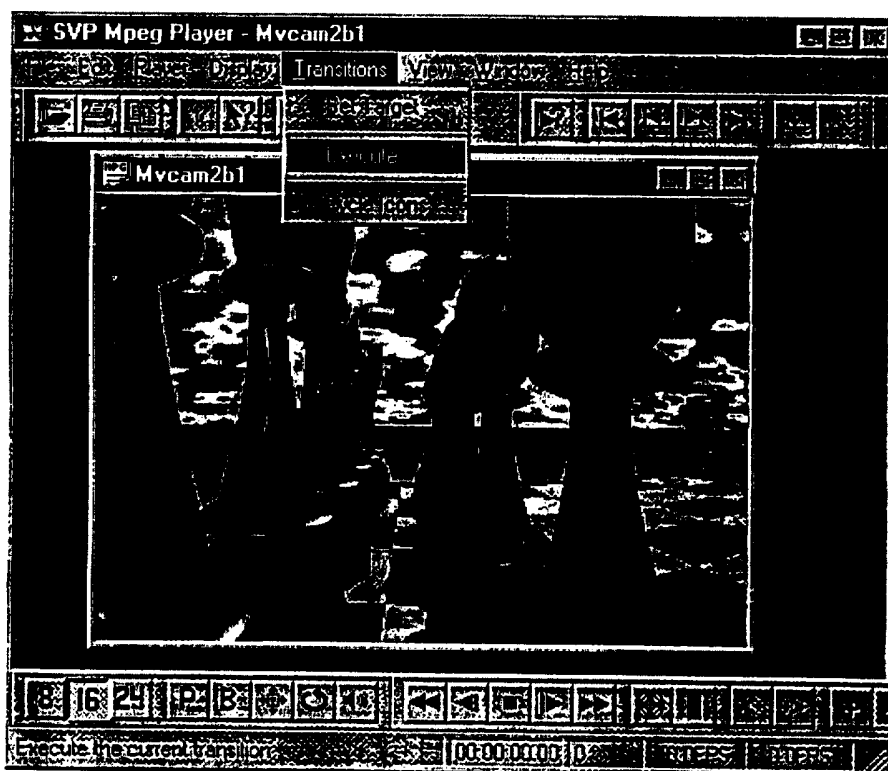
**Primary Window SVP Mpeg Player Player**



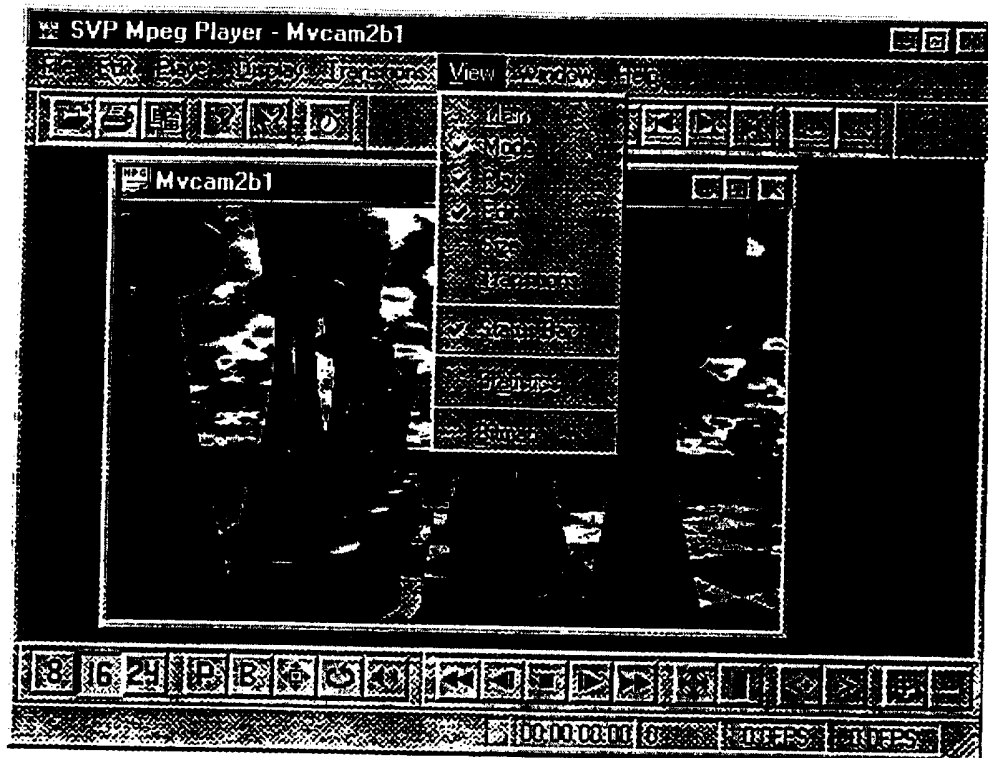
**Primary Window SVP Mpeg Player Display**



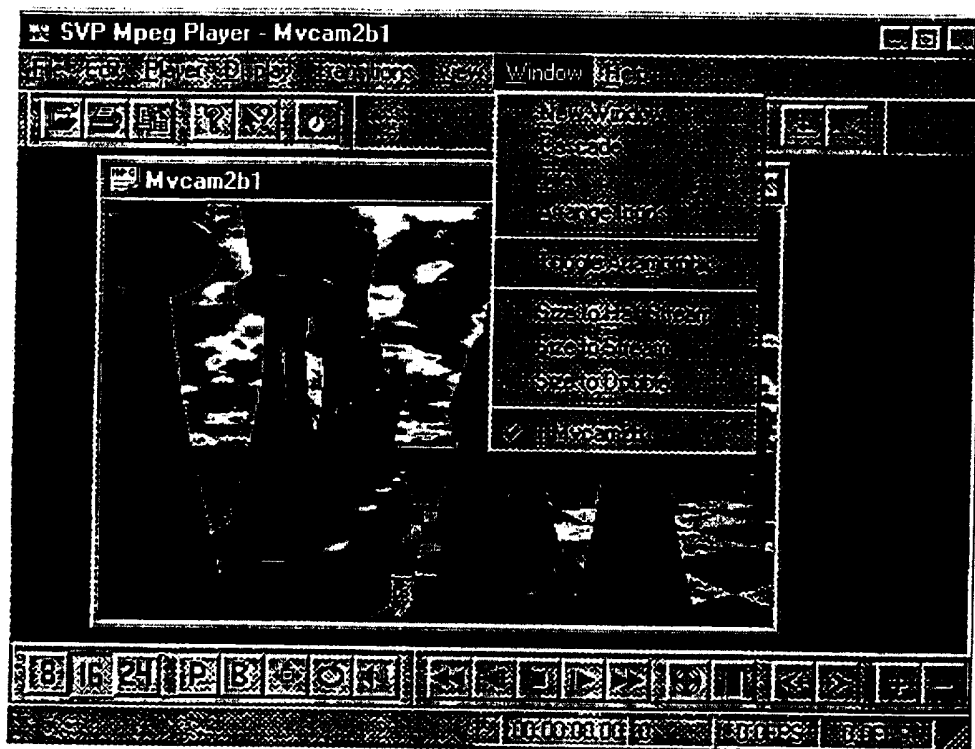
Primary Window SVP Mpeg Player Display Zoom



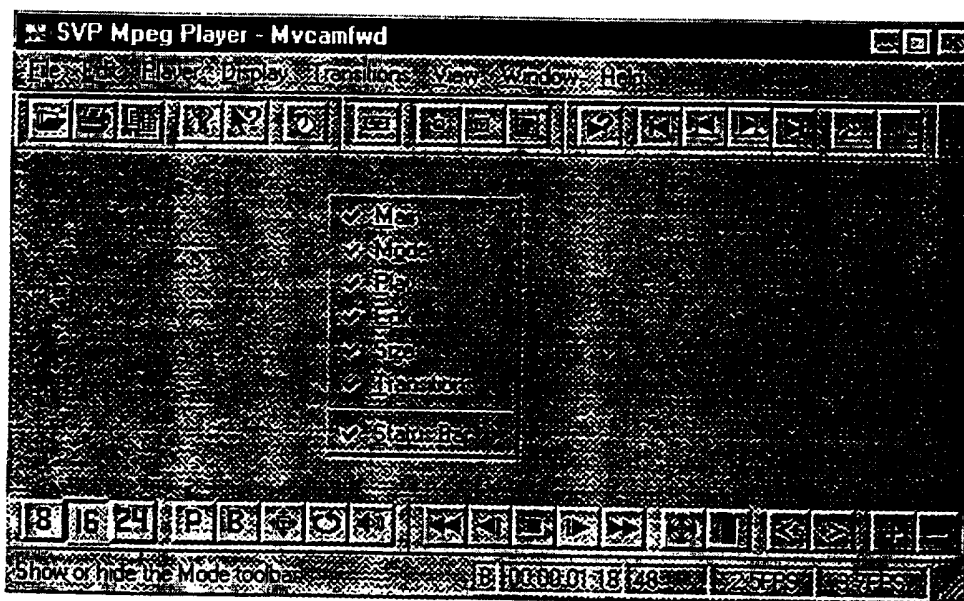
Primary Window SVP Mpeg Player Transitions



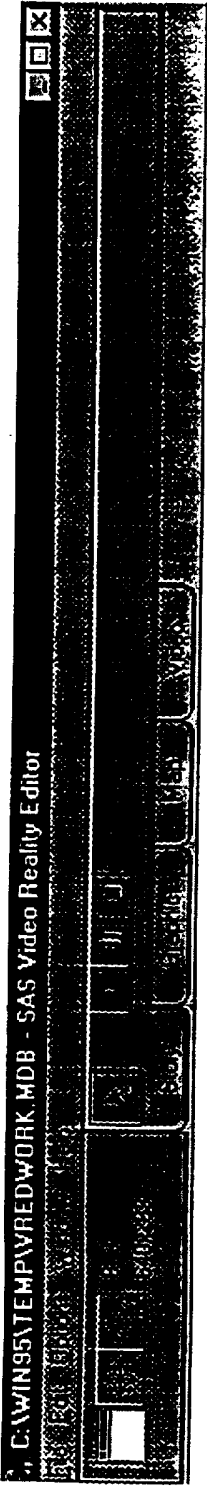
Primary Window SVP Mpeg Player View



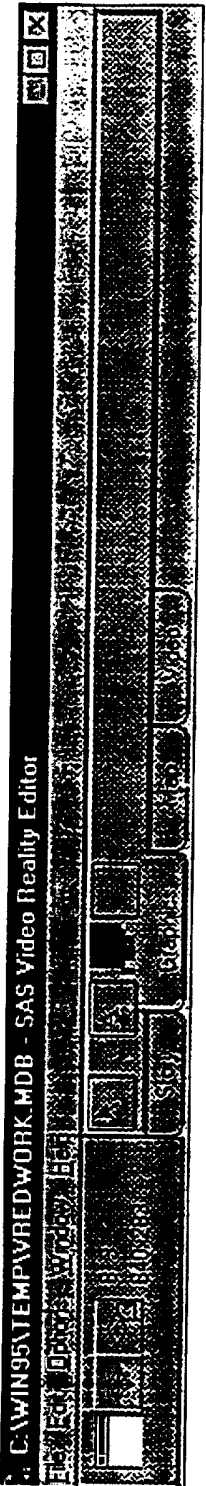
Primary Window SVP Mpeg Player Window



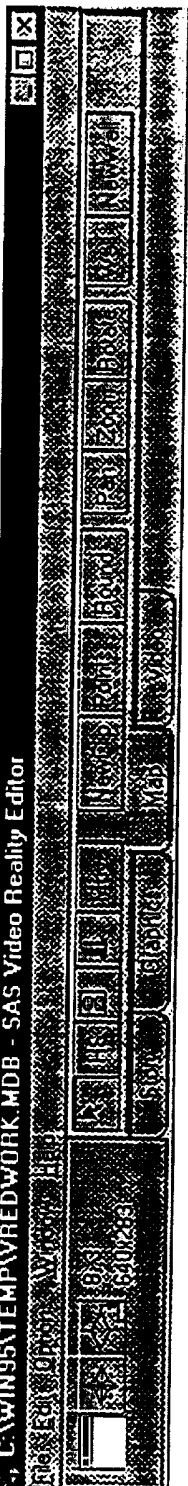
Primary Screen Window SVP Mpeg Player Mouse Right Click



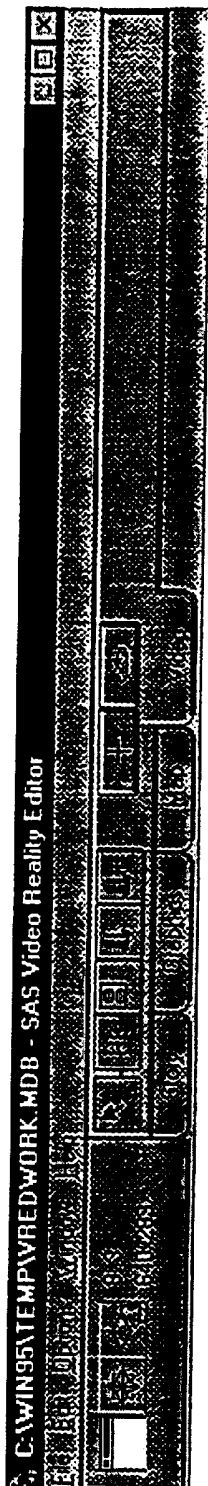
Primary Screen Story



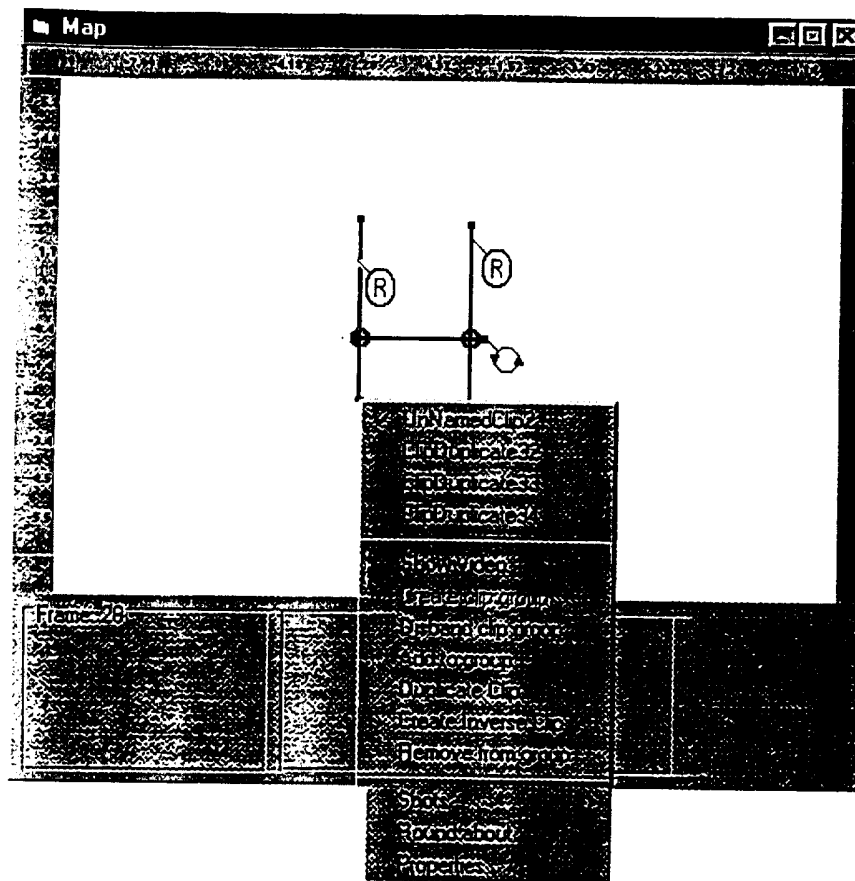
Primary Screen Graphics



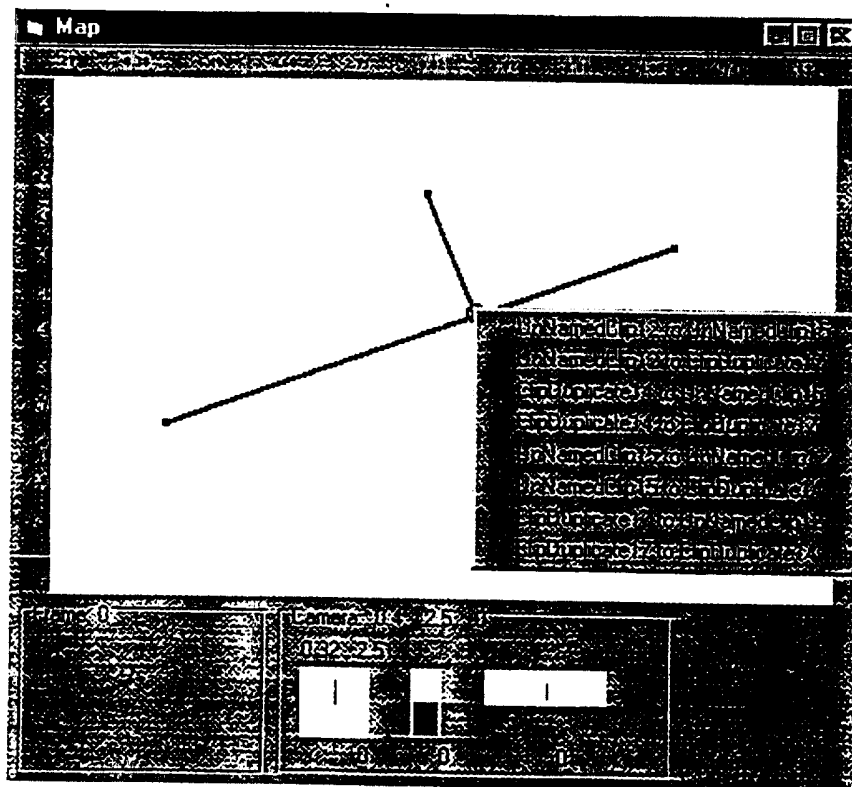
Primary Screen Map



Primary Screen Video

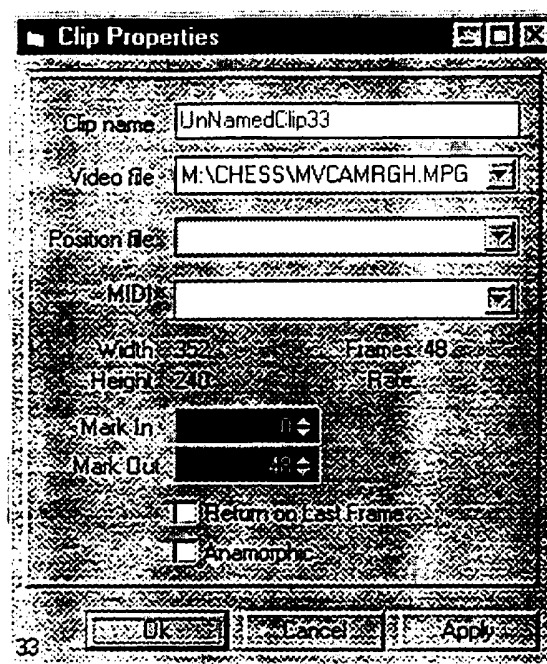


**Primary Window Maps Mouse Right Click on Camera Path**

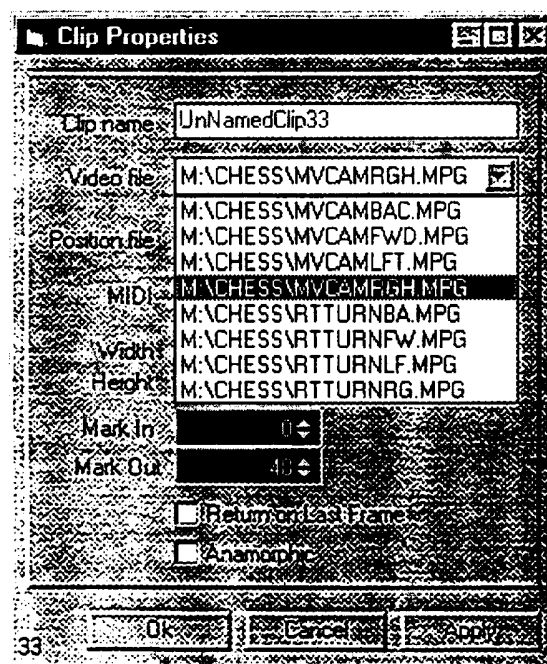


**Primary Window Maps Mouse Right Click on Camera Path Intersection**

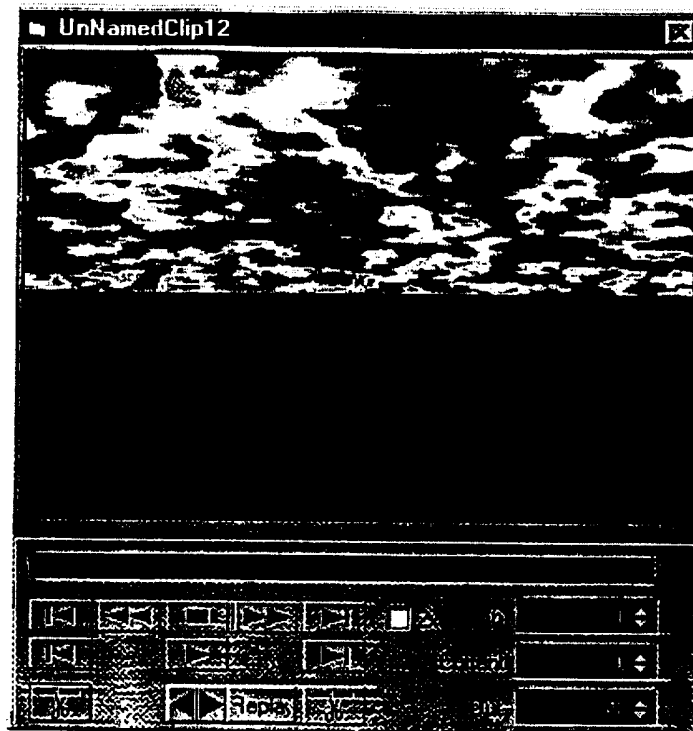




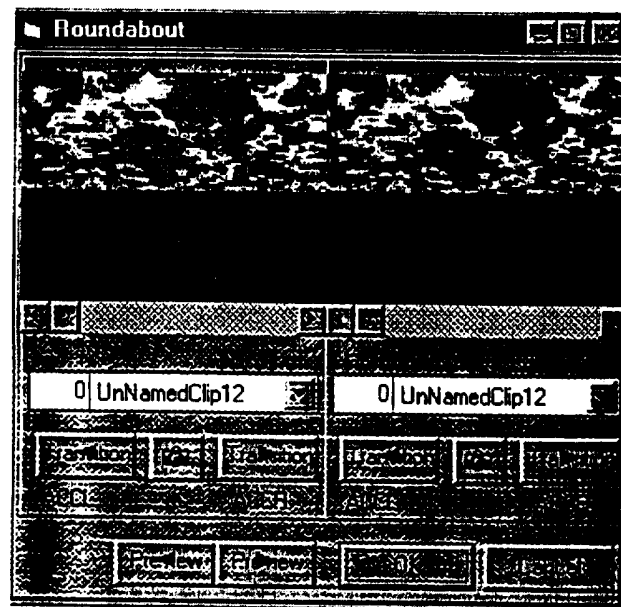
Primary Window Maps Mouse Right Click on Camera Path Properties



Primary Window Maps Mouse Right Click on Camera Path  
Properties Video File down arrow










Primary Window Maps Mouse Right Click on Camera Path Show Video



Primary Window Maps Mouse Right Click on Camera Path RoundAbout



Primary	Window	Maps	Mouse Right Click on Camera Path	Roundabout	Transition	Turn
						

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US97/07359

**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(6) : G06T 5/10

US CL : 395/806

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. : 395/806, 133, 135, 326-328

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)

IEEE CD-ROM

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Chen et al., QuickTime VR-An Image Based Approach to Virtual Environment Navigation, Computer Graphics, JULY 1995, p. 29-38, see p. 29, 31 and 38	1-247
Y	McMillan et al., Plenoptic Modelling: An Image Based Rendering System, Computer Graphics, JULY 1995, p. 39-46, see p. 39, 40, 44 and 45	1-247
Y	US 5,434,592 A (DINWIDDIE, JR. ET AL.) 18 JULY 1995 (18.7.95) see abstract and fig. 1	1-247
Y	US 5,414,801 A (SMITH ET AL.) 09 MAY 1995 (9.5.95) see abstract, figs. 9-13 and cols. 7-20	1-247

☒ 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 07 AUGUST 1997	Date of mailing of the international search report <b>05 SEP 1997</b>
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 <i>Anton W. Fetting</i> ANTON W. FETTING Telephone No. (703) 305-8449

## INTERNATIONAL SEARCH REPORT

 International application No.  
 PCT/US97/07359

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,499,146 A (DONAHUE ET AL.) 12 MARCH 1996 (12.3.96) see abstract, fig. 1	1-247
Y	US 5,495,576 A (RITCHEY) 27 FEBRUARY 1996 (27.2.96) see abstract, figs. 1, 14-16	1-247
Y	US 5,479,597 A (FELLOUS) 26 DECEMBER 1995 (26.12.95) see abstract, fig. 1, cols. 1-9	1-247
Y	US 5,444,478 A (LELONG ET AL.) 22 AUGUST 1995 (22.8.95) see abstract, fig. 7, col. 6-12	1-247
Y	US 5,396,583 A (CHEN ET AL.) 07 MARCH 1995 (7.3.95) see abstract, fig. 2-6	1-247
Y, E	US 5,650,814 A (FLORENT ET AL.) 22 JULY 1997 (22.7.97) see abstract, fig. 3, col. 6-16	1-247
Y,E	US 5,644,694 A (APPLETON) 01 JULY 1997 (1.7.97) see abstract, figs. 1-8, cols. 4-16	1-247
Y,P	US 5,602,564 A (IWAMURA ET AL.) 11 FEBRUARY 1997 (11.2.97) see abstract, figs. 1-7, col. 6-18	1-247
Y,E	US 5,642,477 A (DE CARMO ET AL.) 24 JUNE 1977 (24.6.97) see abstract	1-247
Y,E	US 5,629,732 A (MOSKOWITZ ET AL.) 13 MAY 1997 (13.5.97) see abstract	1-247