PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: **WO 97/26608**

(43) International Publication Date: 24 July 1997 (24.07.97)

(21) International Application Number: PCT/CA97/00039

(22) International Filing Date: 20 January 1997 (20.01.97)

(30) Priority Data:

60/010,214	18 January 1996 (18.01.96)	US
08/597,087	5 February 1996 (05.02.96)	US

(60) Parent Application or Grant
(63) Related by Continuation
US 08/597,087 (CON)
Filed on 5 February 1996 (05.02.96)

(71) Applicant *(for all designated States except US)*: VICOM MULTIMEDIA INC. [CA/CA]; 11603 - 165th Street, Edmonton, Alberta T5M 3Z1 (CA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: LIGHTHEART, Michael, A. [CA/CA]; 70 Glenwood Crescent, St. Albert, Alberta T8N 1X5 (CA). HENDERSON, Scott, R. [CA/CA]; 6624 - 187 Street, Edmonton, Alberta T5T 2N2 (CA). DURNFORD, James, Donald [CA/CA]; 210 Lilac Drive, Sherwood Park, Alberta T8H 1W2 (CA). HEUPEL, Johannes [DE/CA]; P.O. Box 2609, Stony Plain, Alberta T7Z 1Y2 (CA). REDDY, Praveen [CA/CA]; 4020 - 105B Street, Edmonton, Alberta T5T 1V2 (CA).

(74) Agent: BAILEY, Thomas, W.; Oyen Wiggs Green & Mutala, 480 - 601 West Cordova Street, Vancouver, British Columbia V6B 1G1 (CA).

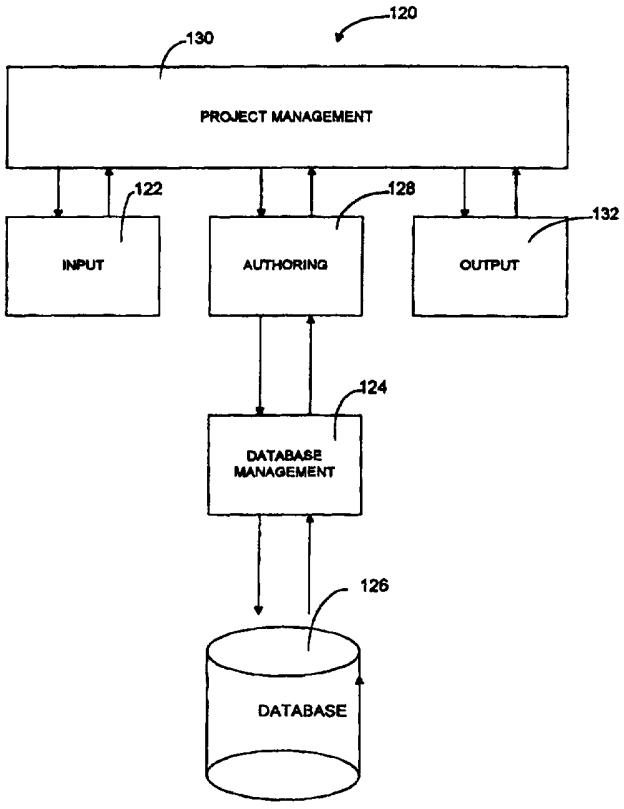
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS,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

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: AUTHORING AND PUBLISHING SYSTEM FOR INTERACTIVE MULTIMEDIA COMPUTER APPLICATIONS

(57) Abstract

A system for authoring and publishing multimedia works has an integrated project management system which controls and tracks the operation of media input, database management, authoring, and output subsystems. The database management subsystem stores all of the elements of an application in a database which eliminates the need to track path names for media elements. Media elements and program elements for multimedia works are stored in the same database and can be made available to all users on a computer network. Multiple authors can simultaneously author a single work.



FOR THE PURPOSES OF INFORMATION ONLY

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

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

AUTHORING AND PUBLISHING SYSTEM FOR INTERACTIVE MULTIMEDIA COMPUTER APPLICATIONS

Field of the Invention

This application relates to a system, including apparatus and computer implemented methods, for authoring, publishing and running computer applications. The invention has particular application to authoring, publishing and running interactive multimedia works, such as interactive tutorials. The system includes a database for storing elements of computer applications and preferably has an integrated project management system which controls and tracks the operation of media input, database management, authoring, and output subsystems.

Background of the Invention

Multimedia works are becoming standard tools for educational, entertainment and business purposes. In the business field, paper-based technical service and training manuals are being replaced by interactive CD-ROM based products. Electronic manuals enable users to access technical information more quickly and efficiently than paper-based materials, while providing enhanced facilities for user-driven training and skills development. While interactive manuals and other CD-ROM based multimedia works offer obvious benefits, producing an interactive multimedia work is time intensive. This makes the cost of production high as compared to paper-based resources.

There is a need for systems for streamlining the multimedia production and post-production process.

The traditional process for authoring and producing multimedia works, such as interactive training manuals, is time-consuming and labour-intensive. The usual first step in such a project is to design the general framework of the desired multimedia application, including scope, content structure and instructional and learning strategies.

The next step is to collect the various physical media elements which will comprise the content of the application. Depending upon the application, the media elements may include text, graphic images, sound clips, film clips and the like. Some media elements may have to be created specifically for the application. The proposed interactions between the various media elements must also be plotted.

Management of media elements is often a severe bottleneck in multimedia production. Before media elements can be combined into a multimedia work they must be made accessible, in computer readable form, to a computer system where they can be combined. Some graphic, text and animation elements may already be available in digital form. Other media elements must be converted to compatible digital files, which must be identified, stored, and then brought individually to the authoring process.

Current methods of multimedia production require users to manually manage media elements both before and after they have been stored in a computer system. Computer readable media elements are typically stored in "file-based" systems which require an operator to assign a file name to each media element. In many computer systems the file names are limited to short, cryptic, combinations of letters and/or numbers. A user can only access a media element in such methods by recalling and using the assigned file name. Further,

the structures and contents of the media elements are often not clear from the file name or from the media elements themselves. This information must be tracked manually. Usually the collection, inventorying, input and digitization of media elements is performed by skilled technicians which inflates production costs.

Traditional authoring systems require a single author (at a single site) to manually link various media elements in order to create a desired multimedia application. The author requires a detailed knowledge of the authoring system which may take months, or years, to acquire. Typically the author is trained in computer programming but is not an expert in the subject matter in question. For example, if the work under development relates to an interactive service manual for use by electronics repair technicians, the author may lack a detailed understanding of the electronic system to which the manual relates. Conversely, repair technicians or other subject matter experts ordinarily lack the training and background necessary to effectively author multimedia applications using existing software. While some simplified authoring tools are available for use by lay persons, they lack the power and flexibility needed to create a full range of multimedia works.

The final step in the production process is to create a run-time application of the work for use by end-users. The program may be stored on a CD-ROM or loaded directly onto a file server hard drive or some other computer-readable medium, depending upon the end-user's requirements.

Post-production revisions and updates often pose a problem. Changes to the interactive storyboard can have a "ripple effect" through the entire work which may necessitate a large amount of re-authoring. If the work is

"media-dense", location and replacement of specific media objects is often a time-consuming and expensive exercise.

The traditional multimedia authoring and publishing process described above suffers from several inherent disadvantages. The linear "one-off" production process is expensive and time-consuming, requiring the cooperation of many specialized people. Although subject matter experts may be involved in the project in an advisory role, they are usually not skilled enough in the use of currently available multimedia authoring software to take part directly in the authoring process. The shareability of media elements and authoring sequences between different projects and applications is restricted. In the case of media dense, intensively interactive applications, file-based media elements can become unmanageable.

Moreover, in the traditional approach, if any project management controls or schedulers are used, they are usually not fully integrated with the production process. This makes it more difficult to eliminate workflow bottlenecks or fast track selected projects. The external, non-integrated project management tools currently available do not effectively allocate personnel and resources between different multimedia projects to enable a true "production-line" approach to creating multimedia works.

Some systems are known in the prior art which are designed to streamline the process of multimedia production. United States Patent No. 5,307,456, MacKay, issued 26 April, 1994 relates to an integrated studio for producing multimedia works such as films or studio for producing videos. The studio includes various multimedia production resources coupled to real-time local area networks. The system provides multiple users with control of a

plurality of dynamically allocated shared resources using a common user interface. The real-time nature of the production system makes it particularly well suited for use in the media production of "live" events such as news events, concerts and sporting events. The user interface enables multimedia elements to be created, edited, bundled, integrated and rearranged by multiple authors. The system includes databases to store some media elements. Data elements may be stored and managed by an object-oriented database which also stores the current state of production resources and routing information.

The MacKay et al system is not tailored for producing interactive multimedia computer applications. It does not provide facilities for assembling or managing computer software. Further, it is not adapted for creating interactive technical manuals and the like. It lacks a scheduling system which is capable of prioritizing different projects.

The use of an object-oriented database to store media elements as disclosed in the MacKay patent is an improvement over traditional file-based storage systems. Other multimedia applications have adopted a similar approach. United States patent No. 5,412,774, Agrawal et al, issued 2 May, 1995 discloses an apparatus and method for storing and retrieving multimedia objects, such as pictorial, textual or audio data, in an object-oriented database. The system includes a display function for displaying objects in each object class. The display functions are also stored in the database. When a user at a system terminal selects a media object for display in a graphical interface, the system processes the media object or object class using the display function associated with the object or object class. An object manager controls access to the database. Objects may include embedded references to other objects which are accessed via the object manager.

The Agrawal et al. system is designed for accessing and displaying media elements from a central database according to a predetermined format which is determined by the design of the database. Agrawal et al do not include an authoring system for creating multimedia works. Users cannot establish an unrestricted number of links or associations between selected media objects.

United States patent No. 5,349,648, Handley, issued 20 September, 1994 relates to an automatic high speed publishing system for creating printed works comprising both text and graphics. The Handley system allows textual data to be entered by unskilled personnel in "assembly-line" fashion. The data entry personnel do not need to concern themselves with text formatting but need only specify the general type of information being entered. The way that different elements in the finished work will be formatted is handled later by a skilled operator. The Handley system is primarily file-based and requires the operator to specify text to be included in a publication by providing computer path names. The Handley system is not well suited to importing diverse media types and is not able to create interactive, media dense, multimedia applications.

There are some systems which enable users of the Internet to download computer programs which are run on their local computer from a server on the Internet. An example of this is Sun's JAVA which is a system for creating applets that can be downloaded to a web browser program running on a local computer. Some disadvantages of these systems are that there is no simple system for creating new applications. The applications are written by computer programmers in traditional ways. Further, if a user wishes to run an application, the entire application is transmitted to the user, not just the part of the application that the user wants to run. These systems can, therefore, cause a lot of network traffic.

Summary of the Invention

An object of this invention is to provide methods and apparatus for creating computer applications which overcome some of the deficiencies of prior art systems.

Another object of this invention is to provide methods and apparatus for authoring interactive multimedia works which overcome some of the deficiencies of prior art systems.

Another object of the invention is to provide a multimedia publishing system having a centralized database for storing media elements and integrated media input, object management, authoring, project management, and output subsystems associated with the database for authoring multimedia works in a production-line manner

Brief Description of the Drawings

In drawings which illustrate embodiments of the invention, but which should not be construed as restricting the spirit or scope of the invention in any way,

Figure 1 is a data-flow diagram of a multimedia publishing system according to the invention;

Figure 2 is a schematic diagram showing a computer network which may be used in the invention;

Figure 3 is a flow chart showing the overall sequence of steps in authoring a multimedia work according to the invention;

Figure 4 is a screen print showing a partial list of tasks required to create a typical multimedia work;

Figure 5 is a block diagram of an input subsystem according to the invention;

Figure 6 is a diagram showing the structure of a media element;

Figures 7A through 7D illustrate the components of four types of program elements which are used to provide program logic in applications created according to this invention;

Figure 8 is a block diagram illustrating the interrelationship between the classes of elements according to a preferred embodiment of the invention;

Figure 9 is a block diagram showing links between elements in a simple exemplary application;

Figure 10 is a view of a computer screen illustrating the appearance of the application of Figure 9 to an end user;

Figure 11A is a schematic illustration showing the relationship of run-time interpreter software to a workstation and Figure 11B is a schematic illustration showing a possible architecture for a run-time interpreter for use in the invention;

Figure 12 is a flow diagram illustrating a sequence of steps performed by a run-time interpreter in running an application;

Figure 13 is a flow diagram illustrating a sequence of steps performed by a run-time interpreter in interpreting process elements in an application;

Figure 14 is a schematic diagram illustrating the relationship of an authoring program to application elements stored in a central database;

Figure 15 is a simplified view of a screen illustrating a user interface in the system of the invention;

Figure 16 is a view of a screen showing the appearance of the user interface of the invention for a newly started application;

Figures 17A through 17D are flow charts illustrating the sequence of steps involved in authoring a simple example application according to the invention;

Figure 18 is a view showing an exemplary properties window for a control element of type button;

Figure 19 is a schematic illustration showing the organization of information in a database containing application elements for use in the invention; and

Figures 20A and 20B are views showing screens of a user interface of an exemplary application.

Detailed Description of Preferred Embodiments of the Invention

Computer systems may be implemented using many different combinations of computer hardware and software. It will be apparent to those skilled in the art that the detailed implementation of the invention will vary depending upon the hardware and software tools used to practise the invention. To avoid confusion and to enable those skilled in the art to practise this invention, the invention is described using illustrative flow charts, screen displays, and algorithmic descriptions. Things which are well known to those skilled in the art, such as programming techniques and details of computer hardware and software are generally not described in detail in the following description to avoid obscuring the invention.

In the following description, the first part of each reference numeral indicates the drawing on which the element identified by the reference numeral

first appears. So, for example, element **120** is shown first in Figure 1. Element **1400** is shown first in Figure 14.

GLOSSARY

Application - an application is a set of computer instructions that causes a computer to behave in a desired way.

Application Element - an element of an application. Application elements include media elements and program elements. All application elements include a field containing a unique descriptor.

App Set - an "App Set" is a set of application elements that are stored together so that they may be retrieved and reused in the authoring of applications.

Author - an author is a person who uses the methods and apparatus of the invention to create new applications.

Media data - media data is a representation in computer readable form, of information which can be perceived by a person playing back a multimedia application. For example, media data could be: a body of data representing an image in a format, such as TIFF, GIF, JPEG etc.; alphanumeric data; a document in a particular word processing format; an animation file; etc. Media data is not meaningful until one understands its content and structure. For example, a user could consider a file containing the data representing a 24-bit scan of a painting to be a meaningless string of ones and zeroes. If a user attempted to play such data as an audio clip then the result would likely be meaningless white noise.

Media element - A media element is media data combined with information which identifies the content and structure of the media data and/or gives other information about the media data or its relationships to other information.

Physical Media - Physical media are any physical items which contain information which can be perceived by a person which can be captured and stored in a computer storage device as media data. For example, physical media include photographs, negatives, sound recordings, video recordings, text documents etc. Physical media can be anything which is not already stored in digital form.

Program Element - those elements of an application that are not media elements. Program elements typically include one or more fields containing instructions instructing a computer to do something in a desired way, information describing the relationship of the program element to other elements in an application and fields which contain information about the program element.

Subroutine - a set of one or more linked application elements which are invoked by a gosub process element and which, when completed, chain to the element pointed to by the gosub process element that invoked the subroutine.

User - a user is a person who runs applications according to the invention.

1. Overview

This invention relates to a system **120** for creating computer applications such as multimedia works and to methods for creating and playing back computer applications. As shown in Figure 1, an exemplary system **120**

includes: a media input system **122** for capturing physical media in computer readable form and cooperating with other systems for associating information with the resulting media data to make media elements; a database management system **124** for cataloguing the elements of multimedia works; an object-oriented database **126** for storing media elements and other elements of multimedia works (as is described in detail below); an authoring system **128**, for linking elements together to create multimedia works; a project management system **130** to control, track and coordinate the operation of other parts of system **120**; and, an output system **132** for outputting finished multimedia works. System **120** facilitates rapid efficient development of highly sophisticated multimedia works.

As shown in Figure 2, system **120** typically includes a number of computer workstations connected to a computer network **270** such as a local area network (LAN) or a Wide Area Network (WAN). Network **270** includes one or more input stations **255** having displays **257**, one or more server computers, **260**, **262** and one or more workstation computers, **272** all connected by network **270**. Typically database management system **124** and its associated database **126** reside in a database server **262**. Project management system **130** may comprise a process running on a separate server **260**. Network **270** may be a peer-to-peer network in which case the functions of servers **260**, **262** are carried out by one or more of workstations **255**.

The steps in a method for developing a multimedia work using system **120** are outlined in Figure 3. When a new project is initiated, basic information about the project, such as the name of the client who the project is for, the working title of the project, the name of the project manager, etc. is input into project management system **130**. If it is known at the outset that certain

physical media will need to be digitized for the project, then the input tasks necessary to digitize these physical media may be scheduled in project management system **130**.

After the project has been commenced, the work is designed. The design phase includes selecting the subject matter of the work, planning the way in which the subject matter will be presented and choosing media content that will be included in the work. Physical media or information which is already in computer readable form and contains the desired media content must then be located and collected. For example, if the work is about travel in Canada then the designer may want to include as media content maps of various Canadian cities, interesting photographs of Canadian scenes, sound clips of Canadian nightlife, text describing places to go and things to do in Canada and video clips. These materials must all be procured. As another example, the project might be to create a multimedia technical service manual for a piece of electronic equipment. The proposed service manual will include a wide variety of media elements including text, technical drawings, video clips, audio and electronic measurements.

After a multimedia project has been designed, the project manager may enter further details about the project into project management system **130** (step **340**). In step **340** additional physical media that need to be digitized and other tasks that need to be completed to finish the project are entered into project management system **130**. The information includes details of the steps which need to be completed to finish the multimedia work being created, and the deadlines for completing those steps. For example, if the project will include two video segments from a videotape, then the start and stop points of each segment are entered into the system. Preferably in step **340** the physical materials, such

as paper drawings, sound recordings, videotapes etc. which will be digitized to obtain the media elements for a project are individually identified by marking them with bar codes or other machine-readable tags.

Figure 4 shows a computer screen display **402** showing a partial list of tasks to be completed for a typical project. Many of the tasks involving digitizing physical media to create media data for inclusion in media elements. Project management system **130** may also be used to request that additional media elements be input through input system **122** after a project has been started.

Next, input system **122** is used to capture the media elements required for the project from the physical media which have been collected and labelled. The captured media elements are stored in database **126** (step **342**). Project management system **130** coordinates data acquisition step **342** and ensures that the resources required to complete the work are made available in time to meet the project's deadlines. In data storage step **344** the digitized data acquired in step **342** is stored in database **126**, together with additional information from project management system **130**, in the form of media elements **680**. Each media element **680** consists of digital data **682** together with a header **684** containing identification and indexing information. After a media element has been registered in database **126**, users of system **120** can locate and retrieve the element by querying database **126**. It is unnecessary for users to know or remember a path name for the media element as is necessary in many prior art systems.

In authoring step **346** and program storage step **348** an author links stored media elements together with program elements to make a multimedia work. The author may be a subject matter expert, or may be a subject matter expert assisted by a technician. As described below, system **120** preferably incorporates an authoring system which has a user interface which simplifies creation of multimedia applications. New elements generated by the author are stored in database **126** in step **348**. As discussed further below, each program element contains one or more fields containing information which can serve as computer instructions and a header containing identification and indexing information.

Finally, when the work has been completed, the finished work is output through output system **132**. Output system **132** may optimize the arrangement of the work to suit the medium in which the finished work will be output.

2. Detailed Description

A. Media Input Subsystem

Input subsystem **122**, shown in detail in Figure 5, is designed to facilitate the automation of media input tasks so that they may be undertaken by non-skilled technicians. Input subsystem **122** comprises a plurality of computer workstations **255** connected to computer network **270**. Each workstation has a display **257** and a bar code reader **558**.

Each input workstation **255** is also connected to one or more input devices such as a scanner **560**, an audio digitizer **561**, a digital oscilloscope **562**,

a videotape player **563**, digital camera **564**. The input devices capture the information content of physical items, such as a videotape, audiotape, or a sheet of paper as media data for storage in database **126**.

The operation of input system **122** is controlled by project management system **130**. Project management system **130** prioritizes pending information capture tasks, identifies which one(s) of workstations **255** are equipped with the input devices needed for each input task and directs each input task to a specific workstation. Input tasks may be routed depending upon what workstations and devices are already in use. Project management system **130** may also allocate certain input tasks to specific human technicians. For example, some technicians accessing the system may be authorized to perform only simple input tasks, but not other more complicated tasks.

When an operator logs into system **120** at one of workstations **255**, project management system **130** prompts the operator to commence the next input task scheduled for that workstation (and that operator). The operator locates the appropriate physical medium and scans its bar code with bar code reader **558**. System **120** verifies from the scanned bar code that the operator has selected the correct physical medium. The operator then inserts the physical medium into the appropriate input device and the system captures the required information in digital form. Preferably technical details, such as the desired output file format, scanning resolution, scanning colour depth, audio sampling rates etc. are set by software under the control of project management system **130** (either according to default values set in project management system **130** or according to specific values selected for that input task by the person who scheduled the task) so that the operator need not be skilled in such matters.

Input subsystem **122** forwards the captured information to database **126** where it is catalogued using information about the structure and content of the media data from project management system **130** and stored as a media element **680**.

System **120** may generate images for display on icons representing media elements **680**. The images may be standard icons representing various types of media element but are preferably low resolution "thumbnail" views of the image represented by media data in each media element. Data representing the "thumbnail" view of a media element may be stored in the record for the media element in database **126**.

B. Database Management Subsystem

System **120** includes a database management subsystem **124** for retrieving, displaying and categorizing elements stored in database **126**. Figure 6 is a block diagram showing the various components of a media element. Each media element **680** includes media data **682**, and a header containing several information fields **684**. The header **684** contain information about media element **682**. Header fields **684** preferably include, at least, the fields shown in Table I.

TABLE I - FIELDS IN RECORD REPRESENTING A MEDIA ELEMENT		
Field Name	Description of Contents	Ref. Numeral
Element Descriptor Field	A unique number (or character sequence) which identifies this element and can be used as a pointer to identify the element. The element descriptor is preferably automatically allocated by database management system **124**. An element can be retrieved from database **126** by requesting the element by its descriptor.	**685**
Media Type	indicates what kind of media data **682** represents and the format in which the data is stored	**686**
Project	indicates the name of the project	**687**
Index	a group of fields that identify the subject matter represented by media data **682**, the author etc.	**688**
Media Description	a group of fields that identify technical details about data **682**. For example, for a media element which represents an image, these fields specify things such as the resolution, colour depth, file format, size etc.	**689**
Media Properties	Additional properties which describe the characteristics of the media elements.	**690**

Database management subsystem **124** manages the contents of database **126** which includes both media elements **680** and program elements **700** for the creation and playback of multimedia works. Media elements **680** are initially registered in database **126** when they are captured by input subsystem **122** as described above. In the alternative, previously digitized information may be directly imported into database **126**. For example, database **126** can accept a previously digitized bitmap image and store it as a media element **680** for use in authoring. When previously digitized information is imported into database **126**,

database management system **124** requires the user to input at least a minimum amount of information for information fields **684** so that the information can be stored as a media element **680**. System **120** can determine some information about a file containing previously digitized media data, such as the file format, the size of a digitized image etc. by studying the contents of the file.

Database management system **124** ensures that required fields are filled in before an element can be stored in database **126**. Additional elements may be created and automatically stored in database **126** during the authoring process, which is described further below.

Some advantages of storing all media elements **680** in database **126** are that database **126** relieves users from manually tracking media elements. Database **126** and database management system **124** make it completely unnecessary for a user to remember cryptic path names to retrieve, play or create links to media elements. A user can locate and retrieve media elements **680** by querying database **126**. Further advantages of storing media elements in database **126** are that the media elements are available to all users on network **270** and that multimedia applications which use the media elements do not need instructions from a user about the content and structure of the media elements because the necessary information is contained in each media element.

Database **126** may be any suitable existing available database. Database **126** may be a relational database or may be built on some other data model. For example, database **126** could be an object oriented database management systems (OODBMS) database, a flat file database, or an indexed sequential access method (ISAM) database. In the currently preferred

embodiment of the invention database **126** is managed by a relational database management system. For example, database management system **124** may comprise Oracle™ or Sybase™ database software. Database **126** may reside entirely on one storage device, or an array of storage devices in one server computer, or may even be distributed across a computer network. While it is not preferable, database **126** could comprise two or more separate databases, each separate database containing different types of elements.

Preferably database **126** contains library elements **1930** as well as the elements of individual applications. Library elements are elements which may be reused across several applications. Combinations of program elements (App Sets) which may be inserted as a unit, into new applications may also be stored as library elements **1930**.

Another advantage of storing the elements of applications in a database **126** is that it is very easy to patch applications which are found to include errors. This can be done by simply replacing any defective elements in database **126** For example, a large application distributed on CD-ROM may contain errors in one or two program elements. The problem can be corrected by supplying users with corrected copies of the defective elements together with a small utility for replacing the defective elements in database **126** with the corrected elements. In most cases the corrected replacement elements will fit on a cheap low capacity storage medium, such as a floppy diskette. Program elements do not need to be stored in any particular order in database **126**. The structure of an application is determined by links between the elements of the application and not by the order in which they are stored. This is very different form conventional computer programs in which the order in which individual

computer instructions are stored can determine the order in which the instructions are processed. Patching an application, as described above, is much easier and much more inexpensive than replacing the entire application by creating a new corrected CD-ROM as might be required with a traditional monolithic computer application.

C. Application Structure

A multimedia application is built by linking together media elements **680** with program elements **700** which control how an end user will be able to traverse the media elements. Program elements **700** are stored in database **126**, with media elements **680**. A completed application consists of a series of media elements **680** linked together by one or more program elements **700**. Each element in an application points to, or is pointed to by, one or more of the other elements of the application.

According to a preferred embodiment of the invention a multimedia application is created by combining five different types of elements, media elements **680**, and four different types of program elements **700**, namely, control elements **702**, process elements **704**, action elements **706** and hyperlink elements **708**. The elements of an application are retrieved, as they are needed, and interpreted by a run-time interpreter **1130** comprising software running on a user's computer.

Program elements **700** define the flow of a finished application and define the ways that an end user can interact with the completed application. A process element **704** is an element that includes one or more computer

instructions that provide procedural logic for applications. For example, a process element **104** could start an application running, create a control element, delete a control element, change the properties of a control element, call a subroutine, create a loop or a branch, etc. A control element **702** specifies the characteristics for a control, which is something that provides interaction with a user, either by providing a way for a user to provide input to the application and/or by providing a way for the application to provide information to a user. A control element **702** can act as a trigger for an event to occur in an application. For example, control elements **702** of different types may be provided to specify a button, a panel, a hot spot, a certain frame of a video clip etc. An action element **706** defines an action which can be performed when a control element is triggered. A single control element may provide several different triggers. For example, a control element of type button may be associated, or "linked", with: a "Click" action element, which responds when a user uses a pointing device, such as a mouse, to place a cursor on the button and then presses a switch on the pointing device; a "DoubleClick" action element, which responds when a user places a cursor on the button and presses twice in quick succession on a switch on the pointing device; a "MouseEnter" action element which responds whenever the cursor is moved into the area occupied by the button control in question; etc. Finally, hyperlink elements **708** relate other elements to each other as is described in more detail below. For example, a hyperlink element **708** of type NEXTPREVIOUSPAGE could be defined to relate sequentially ordered pages to each other, as described below. Illustrative sample of specific types of process elements **704**, action elements **706**, and control elements **702** are given in Tables II, III and IV below.

As shown in Figures 7A through 7D, the structures of the various types of program element **700** are somewhat different from the structure of a media element. Program elements **700** each include a DESCRIPTOR field containing a descriptor **685** which is unique to that element. Each program element **700** also includes pointer fields that identify other elements to which the program element is linked.

A process element **704** (Fig 7A) typically comprises a TYPE field **710**, which contains the type of process that the process element relates to, a CHILD_ID field **712** which contains a pointer to the next process element in an application, a group of fields **714** which contain operands, such as pointers to media elements, which the process element will affect, a CONTROL_ID field **716** which indicates the control element that the process element affects (if any), and a label **718** which contains a name for the process.

A control element **702** (Fig 7B) typically includes a LABEL field **720** which names the control to which it relates, a TYPE field **722** which specifies the type of control to which control element **102** relates, a group of fields **724** which contain operands which determine various properties associated with the control (For example, if a control element **702** is of type "button" then the properties specified in fields **724** may specify the button's size, position, caption, color, an image to display on the button, and so on). A control element may also have a PARENT_ID field **726** which contains the descriptor **685** for another control on which the control in question is located; an APPVEW_ID field **728** which identifies a record in an external database with which the control is associated, and a NEXT_ID field **730** which can be used to create multiple controls in a single operation, as described below.

An action element **706** (Fig 7C) typically includes a field **740** containing a pointer to a control element to which the action element relates, a TYPE field **742** which specifies the action that will trigger the action element **706**, an operands field **744** containing parameters that affect the operation of the action element, such as a parameter which defines specific frames of a video clip which, when played, will trigger the associated action, and a field **746** containing a pointer to a process element which will be executed when action element **104** is triggered.

A hyperlink element **708** (Fig 7D) includes a TYPE field **750** that contains information that identifies what the hyperlink does, and two or more, and preferably three or more, pointer fields **752, 754,** and **756** which include the unique descriptor for elements (of any type) being linked. First pointer field **752**, ID1, contains the information used to locate the desired hyperlink element. A field **758** may be provided to keep additional information about the hyperlink element **708**. Each type of hyperlink element **708** has an associated type of process element that is designed with knowledge of the structure and organization of the hyperlink element.

In the most preferred embodiment of the invention the various elements of a multimedia application are related to one another by means of a model (the "MAJJIS model") which governs what classes of elements an element in any particular class can point to. Figure 8 illustrates the MAJJIS model. As shown in Figure 8, a process element **704** can point to another process element **704** (link **810**) by means of a CHILD_ID pointer, a media element **680** (link **811**) by means of a MEDIA_ID pointer and/or a control element **702** (link **812**) by means of a CONTROL_ID pointer. A control element **702** can point to another

control element **702** (link **814**), or an action element **706** (link **815**), An action element **706** can only point to a process element **704** (link **817**) and a hyperlink element **708** can point to two or more other elements of any type.

Figure 9 shows the elements which make up a short multimedia application **900**. When application **900** is run it displays a form **1000** (Figure 10) bearing an image control **1002** and a button control **1004**. A portion of the image of image control **1002** is defined as a hot spot control **1006**. When the user clicks on hot spot **1006** an audio clip is played by an audio control **1008** (which does not necessarily have any visible manifestation on the user's screen) and the image displayed in image control **1002** changes to a different image. When the user clicks on button **1004** a subroutine is run which plays a video clip and then terminates application **900**.

The first element **920** in application **900** is a start process element **901**. A start process element **901** begins every application. The record for a start process element **901** may contain fields which contain information such as the name of the application, the project that the application belongs to, the name(s) of the author(s) of the application, and so on. The primary function of a start process element **901** is to point to the first program element in an application. A list of the applications in database **126** can be readily obtained by querying database management system **124** for a list of all start process elements **901** in database **126**.

The elements in an application according to the invention can be subdivided into process chains. Every application has a "start" process chain **970** which consists of a start process element **901** and a series of other process

elements which are linked to start process element **901** either directly, or through other process elements **704**. Each process element has a pointer (CHILD_ID) **712** which contains the descriptor **685** for the process element which will be executed next. In Figure 9, the flow of execution of process chains is indicated by solid lines. Other links between elements of application **900** are indicated by dashed lines.

Each process element in a process chain has a CHILD_ID pointer **712** that points to one subsequent process element (unless it is the last process element in a process chain in which case the CHILD_ID pointer **712** contains a value which indicates that there is no subsequent process element). Other process chains are initiated by action elements. For example, the sequence of elements which causes sound clip **954** to play when hot spot control **936** is clicked is a process chain **972** associated with the CLICK action **950** associated with hot spot control **1006**. There can also be stand-alone process chains (or "subroutines") which are invoked by a process element of type GOSUB and which begin with a process element of a type which starts a subroutine.

When an application is executed on a computer, such as one of workstations **272**, then the computer instructions contained in the process elements **704** in the start process chain are executed in sequence. The particular actions that the computer performs in executing these instructions are determined, in part, by the contents of control elements **702** which are also linked to the process elements **704**.

In application **900**, the first element **920** is a start process element **901** which simply points to another process element **922** of a type which creates a

form. Create control process element **922** contains the instruction CREATE_CONTROL which causes the computer on which application **900** is running to create form **1000** within which other controls for application **900** will be displayed.

Process element **922** creates a form control, rather than some other kind of control, because it includes a pointer to control element **924** which is of type "form". Properties of form **1000**, such as its size and color are stored in form control element **924**. It can be appreciated that form control element **924** modifies the effect of the CREATE_CONTROL instruction which is stored in create control process element **922**.

Process element **922** contains a CHILD_ID pointer **712** to another process element **926**. Element **926** is also of the create control type. Element **926** has a CHILD_ID pointer to image control element **928**. When the CREATE_CONTROL instruction contained in process element **926** is executed then image control **1002** is drawn on form **1000**. Properties of image control **1002**, such as its size, position, whether or nor it initially displays an image and, if so, what image are contained in image control element **928**.

Process element **926** points to another process element **930**. Process element **930** is of a type which sets properties for an already existing control. Process element **930** has a pointer **931** to image control **928** (which indicates that it is the properties of image control **1002** as set by image control element **928** that process element **930** will affect) and a pointer to media element **932**. In example application **900** process element **930** sets image control element **928** so that the image of media element **932** will be displayed in image control **1002**. A process

element of type "set property" such as process element **930** could change any properties of image control element **928**.

Process element **930** points to a process element **934** of type "create control" which points to hot spot control element **936** and therefore creates hot spot control **1006**. Process elements **938** and **942** are chained, in turn, to process element **934**. Process element **938** points to audio control element **940** and therefore creates audio control **1008**, which permits playback to a user of sound clips. Process element **942** points to button control element **944** and therefore creates button control **1004**.

When application **900** is run on a computer, as described below, the computer instructions in each of the process elements of start process chain **970** are executed in sequence, with the result that form **1000** and controls **1002, 1004,** and **1006,** as shown in Figure 10, are displayed on a user's computer screen and audio control **1008** is created to permit sound to be played to a user. Application **900** then simply waits for input from a user.

A user can interact with application **900** by clicking on hot spot control **1006** or by clicking on button **1004**. Hot spot control element **936**, which creates hot spot control **1006** contains a pointer to an action element **950** of a type which responds to clicks. If a user clicks on hot spot control **1006** then process chain **972** is triggered. Process chain **972** begins with a process element **952** of type "set property" which contains a pointer **953** to audio control element **940**. Process element **952** also contains a pointer to media element **954** which contains as media data **682** a sound file that can be played back by audio control **1008**. When process element **952** is executed then the "playstate" property of audio

control **1008** is set to play and the "media_id" property of audio control **1008** is set to point to media element **954**. The result is that the media data of media element **954** is played back to the user.

Process chain **972** continues with a second set property process which changes the properties of image control **1002** (as indicated by pointer **957** to image control element **928**) to display the image represented by the media data of media element **958** instead of the image of media element **932**.

When a user clicks on button control **1004** then process chain **974** is performed. The first process element **962** in process chain **974** calls a subroutine **966** which, in this example, plays back a video clip to a user. Process element **962** contains a pointer to process element **964** which is of a type that starts a subroutine **966**. Subroutine start process element **964** contains a pointer to further process elements (not shown) which make up subroutine **966**. When subroutine **966** is over then execution of process chain **974** continues with process element **968** which terminates application **900**.

It can be appreciated that application **900** links together four media elements **932, 954, 958** and the video media element of subroutine **966** by means of a number of process elements.

Preferably the pointers which constitute the links between elements in an application, such as application **900**, are the same as the unique descriptors **685** which identify the elements being linked. That is, each process element has a CHILD_ID field **712** which contains the descriptor **685** for the next process element in the process chain. Each process element which interacts with a control

has a CONTROL_ID field which contains the descriptor **685** for the control element which defines the properties of the control etc. This allows the run-time interpreter software **1130** on a workstation **272** which plays back a multimedia application to request the next process element in a multimedia application, (or an element which contains information which is needed by a process element) by querying database **126** using the unique descriptor **685** for the requested element.

Storing applications in a database **126** provides concurrency, security and integrity within a set of elements and allows simultaneous authoring and playback of applications.

The system described herein also can minimize traffic across a computer network. When an application is played back, elements need only be loaded from database **126** as they are required. Only those parts of an application which a user traverses need to be sent across network **270**. It is possible, but not usually necessary on a fast network **270**, to pre-fetch additional elements from database **126** and to store the additional elements in a buffer until they are needed. Buffering elements of an application may be desirable to increase the performance of the application if network **270** is a slow network. Network **270** may be, for example, the Internet, in which case it is preferable to buffer the elements of applications before playback. Further, as described below, the information contained in program elements is preferably in the form of short higher level commands, rather than relatively very large blocks of compiled computer code.

D. Method for Running Applications

Preferably applications created by authoring system **128** are run by a run-time interpreter **1130**. The run-time interpreter **1130** preferably comprises software running on a workstation **272**. Run-time interpreter **1130** has access to database **126**.

Figure 12 shows the general sequence of steps that take place when run-time interpreter **1130** runs an application. Figure 13 illustrates the general sequence of steps associated with performing a process (step **1220** from Fig 12). It must be emphasized that these diagrams are not meant to definitively illustrate the actual flow of control in run-time interpreter **1130** Those skilled in the art will realize that run-time interpreter **1130** may be implemented in many possible ways in many different programming styles and languages. Figures 12 and 13 are included to aid in understanding the steps involved in running an application according to the invention but should not be construed so as to limit the scope of the invention.

Run-time interpreter **1130** allows a user to select an application to run (step **1212**). Selection step **1212** typically involves querying database **126** to identify start process elements **901** in database **126** which start applications, to which the user has access rights, and which meet other criteria selected by the user. The user can select the desired application by identifying it from the results returned by the query. If the workstation is running the Microsoft Windows™ operating system this is typically done by double-clicking on an icon representing the desired application.

Run-time interpreter **1130** then retrieves the process element **901** that starts the selected application (step **1214**). When it retrieves starting process element **901**, run-time interpreter **1130** retrieves from the CHILD_ID pointer field **712** of starting process element **901** the descriptor **685** for the next process element in the application (step **1216**). Run-time interpreter **1130** then retrieves the next process element in the application by querying database **126** using descriptor **685** (step **1218**), and so-on, until the end of the start process chain for the application has been reached (the last element in the starting process chain is identified by the fact that it does not include a pointer to any subsequent process element).

Run-time interpreter **1130** may perform the actions specified by process elements as the process elements are retrieved from database **126** (as shown in Fig 12). In the alternative, run-time interpreter **1130** may retrieve all of the process elements in a process chain first and perform the actions specified by process elements after the entire chain has been retrieved. Where the latter approach is taken then run-time interpreter **1130** may optionally perform pre-processing steps before the process chain is run. For example, pre-processing steps may include loading media elements into the memory of workstation **72** ready for playback or doing mathematical calculations etc.

As it retrieves each process element, run-time interpreter **1130** reads the TYPE field **710**, which indicates what type of process element has been retrieved, and reads the element's properties and computer commands from the appropriate fields. The computer commands (or "instructions") are preferably subsumed in the element type (i.e. the actions performed by run-time interpreter **1130** are determined by the element type. A separate field for computer

commands is unnecessary). Run-time interpreter **1130** then causes the computer on which it is running (e.g. workstation **272**) to perform the actions indicated by the process element being interpreted.

The TYPE field of a process element can be very short. For example, a TYPE field may be 50 bytes or less in size and can typically be about 10 bytes in size. Preferably run-type interpreter **1130** causes the computer on which it is running to perform many operations in response to each instruction (i.e. the TYPE fields in the process elements **704** serve as high level instructions akin to instructions in a scripting language). This provides advantages over systems in which large volumes of low level computer code must be transferred over a network.

Figure 11 shows one possible architecture for run-time interpreter **1130**. Run time interpreter **1130** includes a process handler **1131** which coordinates running applications and retrieving process elements from database **126**. Process handler **1131** reads type information from each process element which is retrieved from database **126** and then calls a process function **1132** to perform the actions specified by the process element. A separate process function is provided for each type of process element. Some types of process element may instruct the computer to perform different actions depending upon the type of media element that is being pointed to. For example, a create control process behaves very differently depending upon whether it is creating an audio control, an image control, or some other control type. Consequently, the process function **1132** associated with the "create control" process determines what kind of control is to be created (by parsing the TYPE field **722** of the control element **702** pointed to by the CONTROL_ID pointer of the process element **704** being processed) and

calls the appropriate one of a number of control functions **1133** to create a control of the type required.

Preferably run-time interpreter **1130** builds a data structure such as control table **1140** (Fig. 11A) in the computer's memory. Control table **1140** contains a record **1142** for each control which has been created in running an application. For each control, record **1142** contains a list of the control's current properties. When the control is first created run-time interpreter **1130** calls the handler functions **1150** necessary to implement the control (step **1316**). Functions **1150** may be part of run-time interpreter **1130** or may be provided, in part, by the operating system **1120** of the computer on which run-time interpreter **1130** is running.

In the example described above, a separate "create control" process element was provided to create each control in application **900**. Run-time interpreter **1130** may be constructed so that a single create control process element can create multiple controls. This may be done by means of NEXT_ID fields **730**.

For example, an application to create three controls having properties defined by control elements CONTROL1, CONTROL2 and CONTROL3 contains a "CreateControl" process element PROCESS1 having a CONTROL_ID field containing a pointer to CONTROL1. The NEXT_ID field of CONTROL1 is set to point to CONTROL2 and the NEXT_ID field of CONTROL2 is set to point to CONTROL3. The NEXT_ID field of CONTROL3 is blank. When run-time interpreter **1130** interprets Create Control process PROCESS1 it first retrieves CONTROL1 using the pointer in the CONTROL_ID field of PROCESS1.

Run-time interpreter **1130** then creates the control defined by CONTROL1. Next, run-time interpreter **1130** retrieves the contents of the NEXT_ID field from CONTROL1 and uses those contents as a pointer to retrieve CONTROL2. Run-time interpreter **1130** then creates the control defined by CONTROL2. Next, run-time interpreter **1130** retrieves the contents of the NEXT_ID field from CONTROL2 and uses those contents as a pointer to retrieve CONTROL3. Finally, run-time interpreter **1130** creates the control defined by CONTROL3. As the NEXT_ID field of CONTROL3 does not contain a pointer to any other control element then run-time interpreter **1130** continues by processing the process element following PROCESS1.

In many applications several controls are created at about the same time. Creating several controls with a single process element reduces the number of records required to instruct run-time interpreter **1130** to perform a desired function. This, in turn, reduces the traffic on network **272** and increases the speed at which the application can be played back.

If a control is later modified by a set property process, the set property process causes run-time interpreter **1130** to change the record **1142** in table **1140** which relates to the control being modified. Whenever the properties of an existing control are modified, run-time interpreter **1130** runs the control handler function **1150** associated with that control so that the behaviour of the control is updated. For example, if the "playstate" property of the record for a video control in table **1140** is changed from "STOP" to "PLAY" then run-time interpreter **1130** causes the handler function **1150** for that video control to begin playing the video data associated with the media element currently associated with the video control.

Preferably the computer commands in a process element are symbols or tokens which are interpreted by run-time interpreter **1130** in light of the element's properties and the properties of other program elements to which the process element points. For example, if a process element is a process element of type "create control" then run-time interpreter **1130** retrieves the control element which is pointed to by the process element. If the control element is of type "button" then run-time interpreter proceeds to create a button control by causing a button to be drawn on the computer's screen and by setting up a handler to detect and pass on events caused by a user interacting with the control. Run-time interpreter **1130** preferably includes within itself, or can access by making calls to functions of the operating system on the computer, the computer code necessary to do this. For example, for a button control, run-time interpreter **1130** preferably has the capacity to draw a button on the screen of workstation **272** and to animate the button as a user "clicks" the button using a mouse or other pointing device. Run-time interpreter **1130** preferably does not need to retrieve from database **126** the specific computer code needed to implement these features. The size, shape, position, colour, title, and other aspects of the button are determined by the properties fields of the control element which is pointed to by the process element and which is retrieved from database **126** and read by run-time interpreter **1130**.

For example, when application **900** of Figure 9 is run by a run-time interpreter **1130**, run time interpreter **1130** begins by sequentially retrieving the process elements in start process chain **970** from database **126**. Run-time interpreter **1130** then interprets the process elements in process chain **970** beginning with create control process **922**. Run-time interpreter **1130** first determines that process element **922** is a create control process. Run-time

interpreter **1130** then retrieves form control element **924** and determines that element **924** specifies a form control. Run-time interpreter **1130** then writes the properties from element **924** in a record **1142** in table **1140** and creates and runs a handler function **1150** for the form control. The handler function **1150** displays a form on the screen of the user's computer having the properties defined by form control element **924**. In the Microsoft Windows™ operating environment a handler for a form control may be created, for example by calling the **CreateWindow()** system function. Run-time interpreter **1130** then proceeds to interpret process elements **926**, **930**, **934**, **938**, and **942**. After this has been done, table **1140** contains 5 records, one record corresponding to each of form **1000**, image control **1002**, button control **1004**, and hot spot control **1006**. The fifth record corresponds to audio control **1008**. The user's screen appears as shown in Figure 10.

After run-time interpreter **1130** has interpreted element **942** then it simply waits because the CHILD_ID field **712** of element **942** does not point to any other process element. If a user then clicks on hot spot control **936** run time interpreter loads and interprets process chain **972**. When run-time interpreter **1130** receives set property process element **952** from database **126** it first determines that it has received a process of type "set property" and then determines, by reading the CONTROL_ID field of element **952** (indicated by link **953**),that process element **952** sets properties of the audio **1008** control associated with audio control element **940**.

Process element **952** causes run-time interpreter **1130** to set two properties of the control. First it sets the control to point to media element **954**, which contains media data representing a sound clip to be played. Process

element **952** also sets the "playstate" property of the control to "PLAY". It does this by changing the properties for the control in the appropriate record in table **1140**. When run-time interpreter **1130** changes table **1140** it automatically runs the handlers **1150** for any controls affected by the changes so that the changes manifest themselves. Process element **956** causes run-time interpreter **1130** to change the image displayed in image control **1002** in a similar fashion.

It can be appreciated that process chains **972** and **974** do not need to be retrieved from database **126** until a user has initiated execution of those process chains by clicking on hot spot control **1006** or button control **1004** respectively. If a user never initiates execution of a process chain then the elements of that process chain do not need to be retrieved from database **126**, sent across network **270** or processed by run-time interpreter **272** at the user's workstation **272**.

A subroutine may be handled, for example, by preserving the state of run-time interpreter **1130** and then calling another instance of run-time interpreter **1130** to process the subroutine. After the subroutine ends the original instance of run-time interpreter **1130** continues where it left off.

An advantage of the arrangement described herein is that a single application can run on any computer as long as there is a suitable run-time interpreter **1130** running on the computer. For example, some workstations **272** might be PC computers running Microsoft Windows95. Other ones of workstations **272** might be Apple Macintosh computers running the Apple System 7.0 operating system. Still other ones of workstations **272** might be UNIX workstations. The same application could be run on any of workstations **272** as

long as each workstation is running a version of run-time interpreter **1130** which is compatible with the operating system and hardware of the workstation. The run-time interpreter software can readily be customized so that elements agree with the customs of the user interface for the operating system in which run-time interpreter **1130** is running. For example, an Apple Macintosh run-time interpreter **1130** could display button control elements so that they look like the buttons commonly used in Apple Macintosh applications. A run-time interpreter **1130** for the Windows NT operating system could display button control elements so that they look like the buttons commonly used in the Windows NT user interface, and so on.

A further advantage of this arrangement is that the amount of network traffic is minimized. Only those elements of an application which the user plays back are sent across network **270**. The elements are very small because they can contain short higher level instructions instead of blocks of object code or other low level code. Furthermore, run-time interpreter **1130** does not require all of the fields in the elements of an application to play back the application. Traffic across network **270** can be further limited by sending to run-time interpreter **1130** only the contents of those fields from the elements of an application which run-time interpreter **1130** requires to play back the application. Traffic across network **270** can be reduced further still by providing some processing in database management system **124**. For example, if database management system **124** is a client-server database management system then run-time interpreter **1130** can send a single query (in a suitable query language such as SQL in a manner which is well understood to those skilled in the art) which will result in database management system **124** returning the essential fields from all of the elements in a process chain of an application.

The ability to transmit applications to a remote computer on a network with a minimum of network traffic gives the methods of the invention a great advantage over prior art methods known to the inventors. Preferably, where network 270 transmits information in packets of a given size, the essential information form each element of an application is smaller than a network packet. For example, information sent over the Internet using the TCP/IP protocol is typically broken into packets of 1024 bytes. A single TCP/IP packet could, carry four process elements of record size 256 bytes. Other systems which transmit computer code across a network typically require large blocks of code to be transmitted.

E. Hyperlink Elements

Hyperlink elements provide a means for substituting one set of media elements for another set of media elements during execution of an application. Hyperlink elements also provide a means to control the flow of execution of an application and/or to alter the media elements that are played back to a user of an application on the basis of the state of the application. Hyperlink elements can best be explained by means of a simple example.

Suppose an author wants to digitize a printed technical manual and to write an application that simply displays the pages of the manual one at a time. The application will provide two button controls on each page. One button control will change the display to show the next page of the manual. The other button control will change the display to show the previous page of the manual. Such an application could readily be written using the methods described above. However, it would be tedious to do so. The author's starting materials are a large number of media elements, each of which represents one digitized page. For each

page, the author would need to create links to the correct media elements for the next and previous pages. Creating the links might involve creating new button controls for each page. The button controls would point to "click" type action elements which would, in turn, point to process elements of type "set property" which would, in turn, point to the appropriate image element. The authoring could be somewhat simplified with the use of suitable subroutines but would still be unnecessarily complicated.

Many of these tedious steps can be avoided through the use of an appropriate hyperlink element. For example, a "NEXTPREVIOUS PAGE" type of hyperlink element **708** having the structure shown in Figure 7D can be defined. A NEXTPAGE type of process element and a PREVIOUSPAGE type of process element are also created (typically the creation of a new type of process element involves defining a new process function for incorporation into run-time interpreter **1130** and would not be done by an author).

A separate NEXTPREVIOUSPAGE hyperlink element is then created for each page of the technical manual. Each NEXTPREVIOUSPAGE hyperlink element has a field containing a pointer (e.g. the unique descriptor **685**) for the media element containing the previous page, a pointer to the media element containing the current page, and a pointer to the media element containing the next page. These hyperlink elements are stored in database **126**. The preparation of these hyperlink elements can generally be automated as the media elements which contain media data representing the pages of the technical manual generally include an index field containing the page number.

The NEXTPAGE process element operates as follows (the PREVIOUSPAGE process element functions in an analogous manner). The

NEXTPAGE process element contains a pointer to a control element. In this case the pointer is to the image control which displays the page images for a user. When the NEXTPAGE process element is run, run-time interpreter **1130** determines from table **1140** what is the MEDIA_ID property for the image control which is displaying the current page on a user's screen. The MEDIA_ID property points to the media element which is currently being displayed. Run-time interpreter **1130** then queries database **126** for a hyperlink element for which the "current page" pointer is the same as the retrieved MEDIA_ID property. Database **126** returns the appropriate hyperlink element from which the run-time interpreter **1130** extracts the "next page" pointer. Run-time interpreter **1130** then sets the MEDIA_ID property of the image control in table **1140** to point to the media element identified by the retrieved "next page" pointer. When the MEDIA_ID property changes, then run-time interpreter **1130** updates the image displayed by the image control so that the user sees the next page displayed.

With the hyperlink elements described above, authoring the sample application described herein becomes almost trivially easy. The author need only define an image control and two button controls, one having a click action element pointing to a NEXTPAGE process element, the other having an action element pointing to a PREVIOUS PAGE process element.

A difference between hyperlink elements and other types of program elements is that hyperlink elements are generally retrieved by conducting a query against database **126** wherein the form of the query is determined by the state of the application being run. The query preferably searches for hyperlinks which are of a desired type and which contain a desired

value in ID1 field **750**. The searched for value in the ID1 field could, for example, be the current property of a control or the descriptor **685** for a control. If hyperlink elements **708** are always searched for by searching the same fields, e.g. their TYPE and ID1 fields, then a generic query can be defined for locating any hyperlink **708**. Other types of program elements are retrieved from database **126** by requesting them according to their unique descriptors **685**.

Those skilled in the art will readily understand from the foregoing example that hyperlink elements **708** may be used in many contexts. As another example, an application may be a multimedia technical manual. The application may, for example, be a service manual for a piece of electronic equipment which displays a schematic diagram **2030** and/or an image of a circuit board **2014**. The application allows a user to click on any component **2018** shown on the circuit board **2014** or schematic diagram **2030** to produce a pop up menu which allows the user to select various types of information about the selected component **2018**. The application may, for example have the options: "What is?" - which opens a window **2050** (Figure 20B) containing information about selected component **2018**; "Where is?" - which highlights the selected component on schematic diagram **2030** or circuit board image **2014**; "Measure" - which opens a window displaying information about measurements that can be made to that component; and so on.

This functionality can be achieved through the use of hyperlink elements. As an example, the application may create an image control containing schematic diagram **2030** and create hot spot controls **2036** corresponding to components **2018** on the schematic diagram. Each of hot spot controls **2036** is linked to a "Click" action element which link to a chain of process elements. The

process chains include a process element which produces a pop-up window for receiving the user's selection and calls a subroutine which depends upon the users choice.

For example, if the user picks "What Is? from the pop-up menu, a subroutine is called which includes a process element which queries database **126** for hyperlink elements of type "WhatIs" which contain the descriptor for the selected hot spot control. Database management system **124** searches database **126** and returns the requested hyperlink element which also contains a pointer to a process element which begins a process chain for displaying the information that the user requested.

If a user picks "Where Is? from the pop-up menu, a subroutine is called which includes a process element which queries database **126** for hyperlink elements of type "WhereIs" which contain a field containing the component name which is stored in an index field for the selected hot spot control. Database management system **124** then queries database **126** for hyperlink elements of type "WhereIs" which contain the desired component name. Database management system **124** then returns a hyperlink element containing a pointer to a process element chain which highlights the component of interest in another image control. Analogous processes may be used to display other information.

Hyperlink elements may also be used to create applications which can be played back in any one of several languages. For example, an application may be created in which control captions and media elements may be presented in either the English language or the German language. This can be done by referencing all language-specific captions and media through hyperlinks. Two

types of hyperlink would be provided, the a hyperlink of type ENGLISH_LANGUAGE would contain an identifier in its ID1 field and the unique descriptor **685** for a media element containing the desired English language text (or audio or video) in a second pointer field. A hyperlink of type GERMAN_LANGUAGE would contain the same identifier in its ID1 field and the unique descriptor **685** for a media element containing the equivalent desired German language text (or audio or video) in a second pointer field. A pair of such hyperlink elements would be included for each different language-specific caption and each language specific media element. The identifiers would be different for hyperlink elements specifying different language-specific captions or language specific media elements.

All language-specific captions and media would be included through the use of a "SELECT_LANGUAGE" set properties process element which operates as follows. The SELECT_LANGUAGE process element comprises a field containing the identifier contained in the ID1 fields of the pair of hyperlink elements containing references to the appropriate language-specific media elements. Prior to running the application run-time interpreter **1130** would receive language selection information from a user. The language selection information would cause run-time interpreter **1130** to either always retrieve the ENGLISH_LANGUAGE type hyperlink element or to always retrieve the GERMAN_LANGUAGE type of hyperlink element depending upon whether the language selection information specified that the user wished to view the application in English or German. After this has been done, when run-time interpreter **1130** encounters a SELECT_LANGUAGE process element then it queries database management system **124** for the appropriate type of hyperlink element (i.e. ENGLISH_LANGUAGE or GERMAN_LANGUAGE) for which the ID1 field contains the identifier stored in the SELECT_LANGUAGE process

element. Run-time interpreter **1130** then extracts a pointer to the media element (in the desired language) and sets the MEDIA_ID for the control in question to point to that media element. If a control caption is being set then the text for the caption may optionally be stored in the hyperlink element itself. This method can be used to play back applications in any of many languages.

E. Authoring Method and Authoring System User Interface

System **120** has a graphical user interface which allows users to author even sophisticated multimedia works without typing scripts or lines of computer code a hyperlink of type ENGLISH_LANGUAGE would contain the descriptor for a control in its ID1 field and the descriptor for a media element containing the desired English language text (or audio or video) in a second pointer field. or doing other tasks which are traditionally associated with computer programming and which require substantial knowledge about computer programming. The interface allows users to author applications by graphically manipulating icons which represent media elements **680** and program elements **700**.

Multimedia applications are written in a manner somewhat similar to programming in Microsoft™ Visual Basic. The elements of a multimedia application are created by creating new program elements **700** by copying prototype program elements, setting properties of the copies of the prototype program elements, storing the resulting program elements **700** in database **126** and linking the resulting program elements **700** to each other and to media elements **680** from database **126**. Elements are selected by the author(s) from tool boxes or other displays and then are combined in the graphical user interface.

Thus high-level multimedia works can be intuitively created using a visually-driven "point and click" approach. The authoring subsystem is simultaneously accessible by multiple authors located at multiple workstations.

Because authoring system **128** can identify the kind of media stored in a media element, the authoring system can prevent authors from mistakenly linking the wrong type of media element to a process element. For example, the authoring system can prevent an author from trying to display an image in TIFF format using an audio playback process element.

As all of the elements which make up multimedia applications being authored on system **120** are stored in database **126** the elements can be made accessible to all users on network **270**. This makes it possible for two or more authors to simultaneously author a single multimedia application, something which is impractical on prior art systems.

As shown in Figure 14, when an application is being authored on system **120**, a user at one of workstations **272** runs authoring software **1400** on workstation **272**. Authoring software **1400** interfaces with database management system **124** through network **270**. Authoring software **1400** allows a user to record, catalogue and play back new applications. Preferably authoring software **1400** plays back applications by means of run-time interpreter **1130**. All of the elements of the applications are automatically stored in database **126** by authoring software **1400** as the application is being authored. These elements can therefore be made available to other users who have access to database **126**.

Figure 15 shows a screen **1500** from an exemplary user interface for an authoring system for use in the invention. Figure 15 is somewhat simplified for clarity. For example, the icons, which are represented as featureless boxes in Figure 15, preferably bear images so that they can be readily identified and distinguished from one another. In Figure 15, the simple multimedia application **900** of Figure 9 is in the process of being created. Multimedia application **900** is created in an "application area" **1512**. In Figure 15, application area **1512** is a small area on the computer screen. However, an application area can also fill up the entire screen. Application **900** includes the four visible controls which are shown in Figure 10.

The authoring system interface shown in Figure 15 comprises three tool bars, **1520**, **1522** and **1524**. Tool bar **1520** contains icons representing control elements which a user may select for inclusion in a multimedia application, tool bar **1522** contains icons representing a number of media elements, and tool bar **1524** contains icons representing a number of process elements for selection by an author. The tool bars are movable and sizable. The tool bars may optionally be turned off so that the entire application area is visible. Users may also create new tool bars which contain combinations of elements of their choice.

An authoring navigation control bar **1526** allows an author to switch between record and playback modes and to control other aspects of the operation of the authoring system. The user interface also preferably provides a flow diagram window **1528** which illustrates the interrelationship of the various elements of the multimedia application **900** under construction.

When application **900** is loaded by authoring system **1400**, the authoring system reads the start process chain **970** for the application from database **126**. This is done by requesting from database management system **124** the element pointed to by the pointer field of start process element **901** and then requesting the element(s) pointed to by the pointer field(s) of that element, and so on, until database management system **124** has sent all process elements pointed to, directly or indirectly by start process element **901** to authoring software **1400**. Authoring software **1400** maintains a data structure such as table **1402** which lists all of the links between elements in the start process chain **970** for the application. Table **1402** may be the same as, or a superset of, the table **1140** which is used by run-time interpreter **1130**.

When a user begins authoring or editing elements in a process chain in an application then the author can cause authoring software **1400** to send a message to database management system **124** requesting that the records relating to elements in that process chain be locked. This may be done, for example, by writing a value to a field in the first process element in a process chain to signify that the process chain is "locked". When authoring software **1400** retrieves a new process chain from database **126** it checks this field. If the process chain is "locked" then the authoring software will allow the author to display, but not modify, the process chain. This prevents other users from changing that process chain while it is being edited simultaneously by another user. Locking functions may also be provided by database management system **124** in ways which are known in the art.

As an author adds new elements to a the process chain and/or edits elements in the process chain, authoring software **1400** updates table **1402** and,

when new elements are added or existing elements are changed, writes the new elements and/or changes to database **126**. The user can cause authoring software **1400** to refresh table **1402** by rereading from database **126** the process chain which is being edited. In this manner changes which have been made by other authors (if the author has decided not to lock the process chain) can be displayed in flow diagram **1528**.

Tool bars **1520** and **1522** show either a complete set, or a subset, of prototype control elements, prototype process elements and media elements which are stored in authoring software **1400** and/or in database **126**. Users may query database **126** to select elements which may not be displayed on the tool bars. For example, a user could query database **126** to find all hot-spot type control elements for a particular project. The set of control elements **702** of type "HotSpot" in database **126** which match the query are returned in a new tool bar. These control elements may be moved into other tool bars by dragging and dropping them onto the other tool bars.

Tool bar **1524** is a window into a media element management program. The media element management program may be a separate program from authoring software **1400**. Authoring software **1400** runs the media element management program when an author requests media elements. The author can use media elements located by the media element management program by dragging an icon representing a desired media element from the media element management program window **1524** and dropping the icon on the control where the author wishes to use the media element. Media element management program and authoring software **1400** can communicate by exchanging messages between themselves in ways which are well known in the art. For example, when

authoring software **1400** and the media element management program are running under the Microsoft WINDOWS™ operating system then they may communicate by using Windows messaging functions to exchange messages containing the descriptors **685** for media elements selected by a user for incorporation in an application.

Preferably, prototypes for program elements **700** are stored on workstation **272**, where they can be accessed locally by authoring software **1400**. Each prototype has a preset type and a selected set of default properties which may be set to selected values. Preferably authoring software **1400** contains a set of allowed values for each property of each type of program element. Authoring software **1400** prevents an author from setting any property of an element to an illegal value. Tables II, III and IV contain an exemplary list of the types of prototype elements that are available for authoring purposes.

TABLE II - LIST OF PROTOTYPE CONTROL ELEMENTS	
TYPE	DESCRIPTION
Audio Player	plays audio clips
Image	displays images
MPEG Player	plays MPG, AVI, Video data
System	Allows system settings properties and methods to be accessed
Button	A control button
Check Box	A box that can be toggled between checked and unchecked states
Combo Box	a box into which a user can enter text or select predefined text from a pull-down list
Form	a form for placing other controls on

Frame	a frame for placing around other controls
Group Button	a group of buttons only one of which can be depressed at once
Rectangular Hot Spot	an area which is responsive to mouse clicks. A hot spot may be superimposed over an image control or a portion thereof - rectangular
Elliptical Hot Spot	an elliptical hot spot
Polygon Hot Spot	a polygonal hot spot
Poly Line Spot	an area on a computer screen that a user can interact with
List Box	a containing a number of items which may be selected
Option Button	a group of check boxes, only one of which can be checked at once
Panel	used to group things together on a form or ornament a form
Scroll Bar	a scroll bar
Text Box	a box for accepting and/or displaying text
Table (for database access)	window into a database table
Timer	a timer

TABLE III - ACTION ELEMENTS	
Activate	occurs when a form is activated
Change	occurs when the text property of a text control is changed or the IMAGE_ID property of an image control changes
Click	occurs when a mouse button is clicked with the cursor on the control
Close Media	occurs when media is closed on an image control

Deactivate	occurs when another form is activated
DoubleClick	occurs when mouse button is double clicked with cursor on control
Drop Down	occurs when a list drops down
ExternClick	occurs when a user clicks the mouse with the cursor outside of the control
GotFocus	occurs when a control becomes the current control
KeyDown	occurs when a keyboard key is pressed
KeyPress	occurs when a keyboard key is pressed and relesed
KeyUp	occurs when a keyboard key is released
Load	occurs when a control is loaded
LostFocus	occurs when a control ceases to be the currently selected control
MouseDown	occurs when user pushes a mouse button down
MouseEnter	occurs when the mouse cursor enters the active area for the control
MouseLeave	occurs when the mouse cursor leaves the active area for the control
MouseMove	occurs when the mouse cursor is moved
MouseUp	occurs when a user releases a mouse button
Move	occurs when a control is moved
OpenMedia	occurs when a control opens a new media element
Paint	occurs when an image control redraws its display
QueryUnload	occurs before a form unloads (this element can be used to run a process chain to check to ensure that any information from a form that needs to be saved has been saved before the form is closed)

Resize	occurs when a control is resized
Scroll	occurs when the position of a scroll bar is changed
ScrollStart	occurs when a user starts to move a scroll bar
ScrollEnd	occurs when a user stops moving a scroll bar
Select	occurs when text is selected in a text control
Timer	occurs at a set time relative to a trigger event
Unload	occurs when a form is unloaded

TABLE IV - LIST OF PROTOTYPE PROCESS ELEMENTS	
AddProperty	adds a property to a control element
Break	stops execution (for debugging)
CreateControl	creates one or more new controls based upon one or more control elements
CreateTable	creates a table.
DeleteControl	deletes a control
DeleteProperty	resets a property for a control
Delay	waits for an interval
DoEnd	end of a do loop
DoOnAllControls	performs an action in respect of all controls (e.g. moves or scales all controls by a certain amount)
Else	else - branch instruction
ElseIf	else if -branch instruction
EndIf	end if - branch instruction
Execute	runs an external program
Function	calls a function
If	if - branch instruction

Gosub VPO	runs a subroutine
GotoVPO	branches to a specified process element
Label	a label
Method	performs an action against a control i.e. zooms an image in a specified way, moves the control etc.
NOP	no operation
Peek	looks at an entry on a user-defined stack
Pop	pops one entry off of a user defined stack
Push	pushes an entry onto a user defined stack
ProgEnd	ends an application
SetProperty	sets a property of a control
StartPrj	a start element **901**
StartSub	a start element for a subroutine
TypeDef	define a user defined type
TypeUnDef	deletes a user defined type
While	while - branch instruction
Wend	ends a while loop - branch instruction

As shown in Figure 16, when an application is first started, application area **1512** is blank and flow diagram window **1528** shows only a single process element which represents the start process element **901** of the new application. An author can add controls to the application to permit end users to interact with the completed application.

The user interface preferably allows a user to create and place a new control by dragging a desired new control from tool bar **1520** onto application area **1512**. Typically a user would begin by creating a new control element by

clicking on a desired control type from toolbar **1520** and then using the mouse to drag out an area in application area **1512** where a control will be placed. After the control has been placed in application area **1512**, the authoring system displays a window which shows the properties for the control, which the author can set. Initially the properties window shows default properties which are possessed by a prototype element for a control of the type which is being created.

For example, an author could begin to author application **900** of Figure 9 by opening a new application and creating a form control in application area **1512** . The author does this by clicking on a form control icon in toolbar **1520** and then dragging out a rectangular area inside application area **1512**. The position and size properties of the control are set according to the area defined by the user in application area **1512**. The author can set other properties of the form control **1000** by opening a properties window which lists all properties available for form control **1000**. Preferably authoring software **1400** automatically opens a properties window whenever a new control is created.

When the author has completed setting the properties for the control then authoring software **1400** automatically generates "Create Control" process element **922** and form control element **924** and sends these elements to database management system **124** for storage in database **126**.

As each element is saved in database **126** then database management system **124** creates and returns to authoring software **1400** a unique descriptor **685** which unambiguously identifies the record in database **126** which constitutes that element. Authoring software **1400** uses this information to update table **1402**. Also, as each new process element is added to an application, authoring software

1400 sets the CHILD_ID pointer field **712** of the process element which chains to the new element to the descriptor **685** for the new element. This is done by sending a message to database management system **124**.

Figures 17A and 17B illustrate, by way of example, the sequence of steps which take place when an author begins to create the application **900** of Figure 9 by creating a form control and an image control. These steps produce elements **920** through **932** of Figure 9.

An author may load an application which has been previously created for editing. For example, an author may start and begin to author an application **900** as described above. After completing the steps illustrated in Figures 17A and 17B the auhor could save the application. At a later date the author could select the partially completed application **900** and proceed to add the remaining elements of application **900**.

To do this, the author first causes authoring software **1400** to query database management system **124** for a list of the applications which are stored in database **126**. The query returns the start process element for each application in database **126** which matches the query. Authoring software **1400** displays the results of the query in a window. The author can limit the number of applications which are located by the query by restricting the query to applications for a particular project, or applications which are associated with a selected key word.

When the author locates the application which is to be edited the author selects that application by, for example, clicking on the name of the application in the query results window. When an application has been selected,

authoring software **1400** first retrieves all of the elements in the starting process chain for the application and recreates table **1402**. Authoring software **1400** does this by reading the pointer field of start process element **901** (which contains the descriptor **685** for the next element in the application) and requesting the next element in the application using descriptor **685**, as described above, and so on. When table **1402** and flow diagram **1528** have been recreated then the author can continue to author the application or edit properties of the elements which are already present in the application. For example, the author can proceed to add hot spot control **1006**, an audio control **1008**, and button control **1004** to application **900**.

Figure 18 shows a typical properties window for a button control element. By changing properties for a control element the user can change the shape, size, text and other characteristics of the associated control as it will appear to a person running the ultimate multimedia application.

After an author has created controls which will react to user input the author must specify how the controls will react. In the preferred user interface, a user clicks on the control for which he or she wishes to specify an action element. For example, the user could click on control button **1004** of Figure 15. Flow diagram **1528** would then show the action elements associated by default with button **1004**. The user could then select, for example, the CLICK action element for button **1004**. A CLICK action is typically associated with all button control elements. When the CLICK action has been selected then authoring system **1400** clears application area **1512** and the author can author a new process chain which will be executed when a user clicks on button **1004**, as described above.

A distinction between the methods of the invention and other graphical programming tools, such as Microsoft™ Visual Basic, is that, this invention permits all application elements to be recorded in a shared database **126** as an author creates a multimedia application such as application **900**. This means that the partially authored application is available to other users on the network who can be simultaneously authoring the application. An author who is authoring a multimedia application may, at any time, play back the application, to the extent that it has been completed, by selecting the playback button from navigation tool bar **1526**. Preferably authoring software **1400** incorporates run-time interpreter software **1130** so that an author can play back an application and see exactly what a user would see when the application is played back by run-time interpreter **1130**.

It can be appreciated that the user interface described above hides much of the complexity of writing a multimedia application from the user. Also since elements of a multimedia application are all stored in the same database **126** they are accessible by other users. Application elements which have been combined may be reused many times either within the same multimedia application or by other multimedia applications. When a user wishes to reuse components of a multimedia application the user only needs to change the properties of the program elements and/or add or remove program elements to change the functionality of the application. A user can also change the media content of an application, without changing the overall way that the application functions, by simply pointing those elements of the application that point to media elements to alternative media elements.

E. Reusable Elements

At any given time, database **126** may contain several different applications. For example, Figure 19 is a schematic illustration showing the organization of database **126**.

A completed application **1910** (Application A) already authored by a user is recorded in database **126**. Application **1910** comprises a collection of interlinked program elements **700** and media elements **680**. Also recorded in database **126** is a partially completed application **1920** (Application B). Database **126** also contains an element library **1930**. Library **1930** contains commonly available predefined sets of application elements (App Sets) and subroutines which can be used by authors in the authoring of new applications.

Elements from library **1930** can be retrieved by the author of new application **1920** by querying database management system **124** for the descriptors **685** of elements in library **1930** which meet specified criteria. An author may also manually insert the descriptor **685** for an application element to be linked to another application element into the property window for the other application element.

New application **1920** may be made up of newly created application elements, application elements from old application **1910**, elements from library **1930**, or a combination of all of these. New application **1920** may also contain references to external media **1940** such as files stored within a file management system and/or data in an external database **1950**. Unnecessary references to external media should generally be avoided. Database management system **124**

ensures the concurrency and integrity of elements stored in database **126**. The system cannot prevent the problems that can occur if external elements are deleted or renamed and the records of those external elements in database **126** are not updated.

G. Project Management Subsystem

Project management system **130** tracks the status of all of the projects which are taking place on system **120**. At any given time several applications, at various stages of completion, may be on system **120**. These applications between them may comprise hundreds or thousands of media elements as well as hundreds or thousands of program elements. Project management system **130** tracks the amount of work done on each project, including statistics such as the number of images digitized, the amount of time spent by authors in authoring the application, etc.

The progress of scheduled tasks such as the routine digitization of physical media is automatically tracked by project management system **130**. Unscheduled tasks, such as cropping or editing images are preferably also monitored by project management system **130**. This may be done, for example, by providing acces to the software tools and/or computer hardware needed to complete each unscheduled task through the project management system. Before the project management makes available the software tools and/or computer hardware needed to perform a task it runs a "start task" function. The start task function records information about who is performing the task, what time the task is being started, and other project management information and then makes the resources necessary to perform the task available to the person performing

the task. The start task function may also check to ensure that the necessary resources are available for use. After the task has been completed (or after the task is aborted) project management system **130** runs an end task function. The end task function stores a record in database **126** which contains information about whether or not the task was completed, how much time the task took, what person did the task, on what workstation was the task done, what devices were used etc.

The end task function may also store accounting information in database **126**. Tasks may have associated costs which vary from task to task. Project management system **130** can calculate the cost of completing a task from the time taken, the type of task, and/or other project management information. Project management system **130** can therefore keep an up-to-date running tally of the costs incurred in respect of any project as the project is progressing. This can be impossible with other systems in which there is no good way to keep track of all of the work that is being done on a project.

H. Output Subsystem

After an application has been authored then it is immediately available to all users on computer network **270** who have the necessary permissions to access the application. The application may be played back by such users either by running the authoring system, which is described above, or by using a stand alone program which simply runs the application and does not provide any authoring tools. In most cases multimedia applications authored on system **120** will be exported for use on other computer systems or networks.

Typically, for example, a multimedia application will be written to a CD-ROM.

Output system **132** retrieves all of the elements necessary to run the application from database **126**, optimizes the arrangement of those elements for inclusion on a particular output media, and writes the completed application to the desired output media. Optimization includes excluding elements which were needed solely for authoring and managing the production of the work being output, compressing and/or palletizing media elements for efficient retrieval, and removing redundant logic and/or consolidating repeated logic within an application. Output system **132** may also write ancillary files which can be used to install the application on a different computer system and a program for running the installed application.

I. Example - Service Manual

One field in which it is becoming increasingly important to prepare complicated multimedia works quickly and effectively is the field of providing service manuals for complicated equipment. Currently some attempts are being made to create hyper text service manuals through the use of SGML (Standard Generalized Mark-up Language). The process of creating an SGML manual is very labour intensive and requires persons trained in authoring SGML documents. Furthermore, after an SGML document has been created a user cannot view small sections of the document as efficiently as might be desired.

In an exemplary application of this invention, the invention is used to prepare a service manual for a piece of equipment, such as a piece of electronic equipment. The application includes an image control which displays an image

showing components of the part being serviced. For example, as shown in Figure 20, An application may produce a screen display **2000** which includes an image control **2010** showing a picture of a circuit board **2014**. Circuit board**2014** has a number of components **2018** mounted on it. A hot spot control **2020** is associated with each component **2018**. By clicking on hot spots **2020**, a user can view information about corresponding components **2018**. Each component **2018** has a name, such as a part number or other reference number. The hot spot control **2020** associated with each component **2018** includes index fields which include the name of the component so that a program author can easily associate the hot spot controls **2020** with the components **2018** to which they relate.

Screen display **2000**includes a nother image control **2030** which displays a schematic diagram **2032** for the circuit of circuit board **2014**. The components represented on schematic diagram **2032** also have hot spots **2036**. Hot spots **2036** also include index fields which include the names of the components to which they relate.

An author can readily author an application which, when a user clicks on one of hot spots **2020** on circuit board **2014**, highlights the hot spot **2036** on schematic diagram **2032** which corresponds to the same component **2018**, or vice-versa. This may be done by linking the "Click" action of hot spot controls **2036, 2020** with a process which runs a method which retrieves the component name from the hot spot control and then locates all other hot spot controls which are indexed as belonging to the same component by scanning table **1140**. The method then highlights all hot spot controls **2036, 2020** which are associated with that component **2018** (by setting the appropriate properties in table **1140**) and makes all other hot spot controls non-highlighted.

Unlike many traditional authoring systems the author does not need to create a separate process or define separate links for every component. A single process can be reused without modification for any number of components. All that is required is that the control elements which define the hot spot controls contain a field accessible to the process which identifies the components to which the hot spot controls relate.

It can readily be appreciated by those skilled in the art that while this invention has particular application to interactive multimedia applications, the methods and apparatus of this invention may be used to create and play back computer applications of any type.

As will be apparent to those skilled in the art in the light of the foregoing disclosure, many alterations and modifications are possible in the practice of this invention without departing from the spirit or scope thereof. Accordingly, the scope of the invention is to be construed in accordance with the substance defined by the following claims.

WHAT IS CLAIMED IS:

1. An authoring system comprising a computer workstation for authoring a computer application by creating program elements and storing said program elements in a database accessible to said authoring system, said authoring system comprising:

   (a) means for creating said program elements, said program elements each comprising an instruction for processing by a computer;

   (b) means for setting properties of said program elements;

   (c) means for storing each of said program elements in said database as a database record and receiving from said database a descriptor uniquely identifying said database record corresponding to each of said program elements; and

   (d) linking means for creating links between said program elements to create said application by storing in a pointer field of each of said program elements said descriptor for one or more other ones of said program elements.

2. The authoring system of claim 1, wherein said database stores a plurality of media elements, each of said media elements having a unique descriptor, and wherein said linking means comprises means for linking said program and media elements by storing in a media identification field in selected ones of said program elements said descriptor of selected ones of said media elements.

3. The authoring system of claim 2, comprising play-back means for running said application, said play-back means comprising:

(a) means for querying said database to retrieve elements belonging to said application;

(b) means for retrieving from said database a program element of said application;

(c) means for interpreting and performing said computer instruction in said program element;

(d) means for extracting information from said pointer field of said program element to identify a next-in-sequence program element in said application; and,

(e) means for querying said database to retrieve said next-in-sequence program element from said database.

4. The authoring system of claim 3, wherein said workstation and said database are connected via a computer network, said computer network comprising means for exchanging information between said workstation and said database in data packets, said data packets having a packet size, and wherein a record size of each of said program elements retrievable from said database is less than said packet size.

5. The authoring system of claim 1, comprising means for creating and updating a data structure accessible to said authoring system, said data structure containing records of the descriptor, properties and pointer fields of each of said program elements comprising said application.

6. The authoring system of claim 5, further comprising a graphical display means for graphically displaying the contents of said table.

7. The authoring system of claim 6, comprising a graphical interface, said graphical interface comprising a first toolbar comprising icons representative of available types of program elements.

8. The authoring system of claim 7, wherein said authoring system further comprises means for displaying a window containing icons representative of said media elements stored in said database or a subset thereof.

9. The authoring system of claim 1, wherein said program elements comprise process elements, said computer instruction in each of said process elements providing procedural logic for said application.

10. The authoring system of claim 9 wherein said process elements comprise create control process elements for which said computer instructions instruct a computer to create or activate control means for providing interaction with a user.

11. The authoring system of claim 10 wherein said program elements comprise control elements, each of said control elements specifying properties for one or more of said control means created or activated by said create control process elements, and wherein each of said create control process elements comprises a control identification field containing said descriptor for one of said control elements.

12. The authoring system of claim 11, comprising input means for receiving from a user information about a control to create or activate, said information including control type information and control properties information, and means coupled to said input means for: creating and

storing in said database a create control process element comprising said control type information and a control element comprising said control properties information; receiving from said database identification information about said control element; and storing said identification information in said create control process element in said database.

13. The authoring system of claim 12, wherein said program elements comprise action elements, said action elements linked to one or more of said control elements, each of said action elements comprising a field containing said unique descriptor for a process element to be executed upon the occurrence of a specified trigger event.

14. The authoring system of claim 13, wherein said application comprises one or more discrete process chains each of said process chains comprising a series of one or more linked program elements, each of process chains commencing with one of said process elements or action elements.

15. The authoring system of claim 14, comprising graphical display means for graphically displaying said process chains during authoring or playback of said application or portion thereof.

16. The authoring system of claim 9, wherein said program elements are each selected from the group consisting of process elements, control elements, action elements and hyperlink elements.

17. The authoring system of claim 16, wherein the permitted links between said process, control, action and hyperlink elements are governed by the rules illustrated in Figure 8.

18. The authoring system of claim 5 comprising means for extracting information representing a current property of a control; means for querying said database for hyperlink elements which have a specified type and which contain a field containing said current property of said control; means for retrieving a hyperlink element located by said query; means for extracting a pointer to a process element from said retrieved hyperlink element; and means for retrieving said process element from said database using said pointer.

19. The authoring system of claim 5 comprising means for extracting information representing a current property of a control; means for querying said database for hyperlink elements which have a specified type and which contain a field containing said current property of said control; means for retrieving a hyperlink element located by said query; means for extracting a pointer to a media element from said retrieved hyperlink element; and means for retrieving said media element from said database using said pointer.

20. A computer system for authoring a multimedia application, said computer system comprising:

(a) a computer network comprising a plurality of computer workstations and a database accessible from said workstations;

(b) a plurality of media elements stored as records in said database, each of said media elements comprising media data; and

(c) an authoring subsystem running on at least one of said workstations for creating program elements and storing said program elements as records in said database,

wherein each of said records in said database comprises a unique descriptor and said authoring subsystem comprises;

(i) means for creating program elements said program elements each comprising an instruction for processing by a computer;

(ii) means for setting properties of said program elements;

(iii) means for storing each of said program elements in said database as a database record and receiving from said database a descriptor uniquely identifying said database record corresponding to each of said program elements; and,

(iv) linking means for creating said application by creating links between said program elements and said media elements by storing in a pointer field of each of said program elements said descriptor for one or more other ones of said program elements or one of said media elements.

21. The computer system of claim 20, wherein said authoring subsystem comprises play-back means for running said application or a portion thereof, said play-back means comprising:

(a) means for querying said database to locate said application or portion thereof;

(b) means for retrieving from said database a program element of said application;

(c) means for interpreting and performing said computer instruction in said program element;

(d) means for extracting pointer information from said pointer field of said program element, said pointer field information identifying a next-in-sequence program element or media element in said application; and

(e) means for retrieving said next-in-sequence program element or media element from said database by querying said database with said pointer information.

22. The computer system of claim 21, wherein said computer network comprises means for exchanging information between said workstations and said database in data packets, said data packets having a packet size, and wherein a record size of each of said program elements retrievable from said database is less than said packet size.

23. The computer system of claim 21, wherein said authoring subsystem comprises means for creating and updating a data structure accessible to said authoring subsystem, said data structure containing records of the descriptor, properties and pointer fields of each of said program elements comprising said application.

24. The computer system of claim 21, comprising authoring subsystem software running simultaneously on two or more of said workstations thereby allowing concurrent authoring of said application by two or more users.

25. The computer system of claim 20, further comprising an output subsystem for retrieving said media elements and said program elements comprising said application from said database and storing said media and program elements on a computer readable storage medium.

26. The computer system of claim 20, further comprising a program management subsystem comprising:

(a) means for storing a list of said media elements required for a multimedia authoring project;

(b) means for querying said database to determine which of said media elements are stored in said database; and

(c) means for scheduling the input of any of said media elements not present in said database.

27. The computer system of claim 26, further comprising an input subsystem comprising:

(a) an input device connected to at least one of said workstations for digitizing physical media or a portion thereof;

(b) means for applying a machine readable identifier containing an identification code to said physical media or a portion thereof;

(c) means at said one of said workstations for reading said identification code from machine readable identifier;

(d) said program management subsystem comprising means for requesting said input subsystem to digitize a specific item of physical media identified by a specific identification code;

(e) means for reading said machine readable identifier on an item of physical media and, if said identification code in said machine readable identifier matches said specific identification code, digitizing said physical media or portion thereof using said input device to create a computer readable media data record; and

(e) means for storing said media data record and said identification code in said database as a media element.

28. The computer system of claim 27, wherein said input subsystem further comprises means for automatically generating type information about said

media data record and storing said type information in said media element in said database.

29. The computer system of claim 20, further comprising an input subsystem for digitally capturing media elements from external devices, automatically generating type information about said media elements, associating index information with said media elements, and storing said media elements together with said type information and index information as media elements in said database.

30. A computer implemented method for authoring a computer application on a computer workstation by creating program elements to link media elements stored in a database, said method comprising the steps of:

(a) processing author provided information, said information specifying instructions to include in said application, and using said information to create new program elements said program elements each comprising an instruction for processing by a computer;

(b) storing said program elements in a database as database records and receiving from said database a descriptor uniquely identifying each said database record corresponding to each of said program elements; and

(c) linking said program elements to each other by storing in said program elements said descriptors for one or more other ones of said program elements;

(d) linking one or more of said media elements to one or more of said program elements by storing in said one or more program elements said descriptors for said one or more media elements.

31. The method of claim 30, further comprising the step of creating and storing in a memory means accessible to said workstation a data structure containing a record of the identity, properties and pointer fields of each of said program elements and media elements of said application.

32. The method of claim 31, further comprising the step of visually displaying the contents of said data structure at said workstation.

33. The method of claim 32, further comprising the step of automatically updating said data structure when said program elements or said media elements comprising said application are altered.

34. The method of claim 30 wherein said instruction is an instruction for execution by a run-time interpreter.

35. A computer implemented method for authoring and storing computer applications, said method comprising the steps of:

(a) providing a plurality of computer workstations and a database;

(b) storing a start element in said database, said start element containing a pointer field;

(c) creating a new element for said application, said new element comprising a computer instruction and information specifying one or more properties;

(d) saving said new element in said database;

(e) generating a unique descriptor for said new element;

(f) linking said new element to said start element by setting said pointer field for said start element to said descriptor for said new element;

(g) creating subsequent elements for said application, saving said subsequent elements in said database, and generating a unique descriptor for each said subsequent element; and

(h) linking said subsequent elements to other elements in said application by setting pointer fields in said other elements to point to said unique descriptor for at least one of said subsequent elements.

36. A computer implemented method for running a computer application at a workstation connected to a computer network, said method comprising the steps of:

(a) providing a computer application stored in a database connected to said computer network, said application comprising a plurality of elements stored as records in said database, each of said elements comprising: a unique descriptor, a pointer field containing a pointer to a subsequent element, and one or more instructions to be performed by a computer;

(b) in a computer processor in a computer connected to said network, running said application by the steps of:

(i) requesting from said database a first element of said application;

(ii) extracting said pointer from said pointer field of said first element;

(iii) requesting from said database, using said pointer, a subsequent element pointed to by said pointer of said first element;

(iv) extracting from said subsequent element a pointer to another subsequent element and extracting from said subsequent

element said one or more computer instructions of said subsequent element;

(v) interpreting and performing said instructions from said one or more subsequent elements;

(vi) requesting from said database, using said pointer from said subsequent element, another subsequent element pointed to by said pointer of said subsequent element; and

(vii) repeating said steps (iv) through (vi) for each of said subsequent elements.

37. The method of claim 36 wherein, for one or more of said elements, said step of interpreting and performing said instructions comprises creating or activating a control, said control comprising means for presenting information to a user of said application and/or receiving information from a user of said application.

38. The method of claim 37 wherein said control comprises control properties, said control properties affecting behaviour for said control.

39. The method of claim 38 wherein, for one or more of said elements, said step of interpreting and performing said instructions comprises, querying said database for a hyperlink element having a specified type, said hyperlink element comprising a field containing a current control property for said control and a field containing a pointer to an element in said database.

40. The method of claim 39 further including the step of receiving language selection information in said computer, said language selection

information indicating whether to run language-specific parts of said application in a first language or a second language.

41. The method of claim 40 wherein said database contains first and second types of language selection hyperlink elements, said first type of language selection hyperlink elements comprising a pointer to a media element in a first language, and a second type of language selection hyperlink element comprising a pointer to a media element in a second language, wherein, for said one or more of said elements, said step of interpreting and performing said instructions comprises, querying said database for a language selection hyperlink element of said first type if said language selection information indicates said first language and querying said database for a language selection hyperlink element of said second type if said language selection information indicates said second language.

42. The method of claim 37 wherein said instructions are not more than 50 bytes in length.

43. A method for running an application on a computer, said application comprising a plurality of process elements stored in a database, said process elements each comprising a computer instruction, a unique descriptor, and a pointer to a next in sequence process element, said method comprising the steps of:

(a) querying a database to retrieve a first process element;

(b) retrieving from said retrieved process element said unique descriptor for a next in sequence process element in said application;

(c) querying said database using said unique descriptor for said next in sequence process element to retrieve said next in sequence process element;

(d) repeating said steps (b) and (c) for subsequent process elements in said application; and,

(e) extracting and performing said computer instructions from said retrieved process elements.

44. The method of claim 43 wherein, for one or more of said process elements, said step of extracting and performing said computer instructions comprises creating or activating a control, said control comprising means for presenting information to a user of said application and/or receiving information from a user of said application.

45. The method of claim 44 wherein said control comprises control properties, said control properties affecting a behaviour of said control.

46. The method of claim 45 wherein, for one or more of said elements, said step of extracting and performing said computer instructions comprises, querying said database for a hyperlink element having a specified type, said hyperlink element comprising a field containing a current control property for said control and a field containing a pointer to an element in said database.

47. The method of claim 46 comprising the step of maintaining a data structure accessible to said computer processor, said data structure comprising a record of said current control property, wherein said step of querying said database for a hyperlink element having a specified type

comprises reading said current property for said control from said data structure.

48. The method of claim 47 further including the step of receiving language selection information in said computer, said language selection information indicating whether to run language-specific parts of said application in a first language or a second language.

49. The method of claim 48 wherein said database contains first and second types of language selection hyperlink elements, said first type of language selection hyperlink elements comprising a pointer to a media element in a first language, and said second type of language selection hyperlink element comprising a pointer to a media element in a second language, wherein, for said one or more of said elements, said step of extracting and performing said computer instructions comprises, querying said database for a language selection hyperlink element of said first type if said language selection information indicates said first language and querying said database for a language selection hyperlink element of said second type if said language selection information indicates said second language.

50. The method of claim 49 wherein said instructions are not more than 50 bytes in length.

51. A system for playing back a computer application in one of two or more languages, said system comprising:

(a) a database containing a computer application, said computer application comprising a plurality of elements stored as records in said database, said elements comprising at least a plurality of first

language language-specific media elements, a corresponding plurality of second language language-specific media elements, a plurality language selection hyperlink elements comprising a plurality of first language selection hyperlink elements each comprising information for retrieving one of said first language language-specific media elements, and a corresponding plurality of second language selection hyperlink elements, each comprising information for retrieving a corresponding one of said second language language-specific media elements from said database;

(b) a computer connected to said database, said computer comprising:

(i) control means for playing media data in said media elements to a user;

(ii) means for receiving language selection information from a user; and;

(iii) means for running an application, said application comprising computer instructions to cause said control means to play said language specific media elements to a user by the steps of:

(1) querying said database for a language selection hyperlink elements of a type specified by said language selection information and having an identifier specified in said application;

(2) receiving from said database information from said hyperlink element, said information including at least information for retrieving one of said first or second language language-specific media elements;

(3) querying said database using said information to retrieve said one of said first or second language language-specific media elements; and,

(4) playing media data from said media element on said control means.

52. A computer readable storage medium containing computer readable instructions for causing a general purpose computer to run a computer application stored in a database accessible from said general purpose computer, by the steps of:

(a) querying said database to retrieve a first process element from said database;

(b) retrieving from said retrieved process element a unique descriptor for a next in sequence process element in said application;

(c) querying said database using said unique descriptor for said next in sequence process element to retrieve said next in sequence process element;

(d) repeating said steps (b) and (c) for subsequent process elements in said application; and,

(e) extracting and performing said computer instructions from said retrieved process elements.

53. A method of generating and storing media elements in a database connected to a computer network, said method comprising the steps of:

(a) storing in a computer a list of items of physical media to be digitized together with indexing information relating to said items of physical media;

(b) generating a unique identification code for each said item of physical media;

(c) creating a machine-readable tag containing said unique identification code for attachment to each of said items of physical media;

(d) scheduling an input task for digitizing each said item of physical media;

(e) sending an instruction to an input workstation connected to said network to perform said input task;

(f) reading said machine readable tag at said input workstation and verifying that said unique identification code matches said item of physical media for said input task;

(g) if said unique identification code matches said item of physical media for said input task, performing said input task by digitizing said physical media or selected portion thereof to create computer readable media data; and,

(h) storing said media data, and said unique identification code in said database as a media element.

54. The method of claim 53, comprising the step of automatically storing in each of said media elements said indexing information.

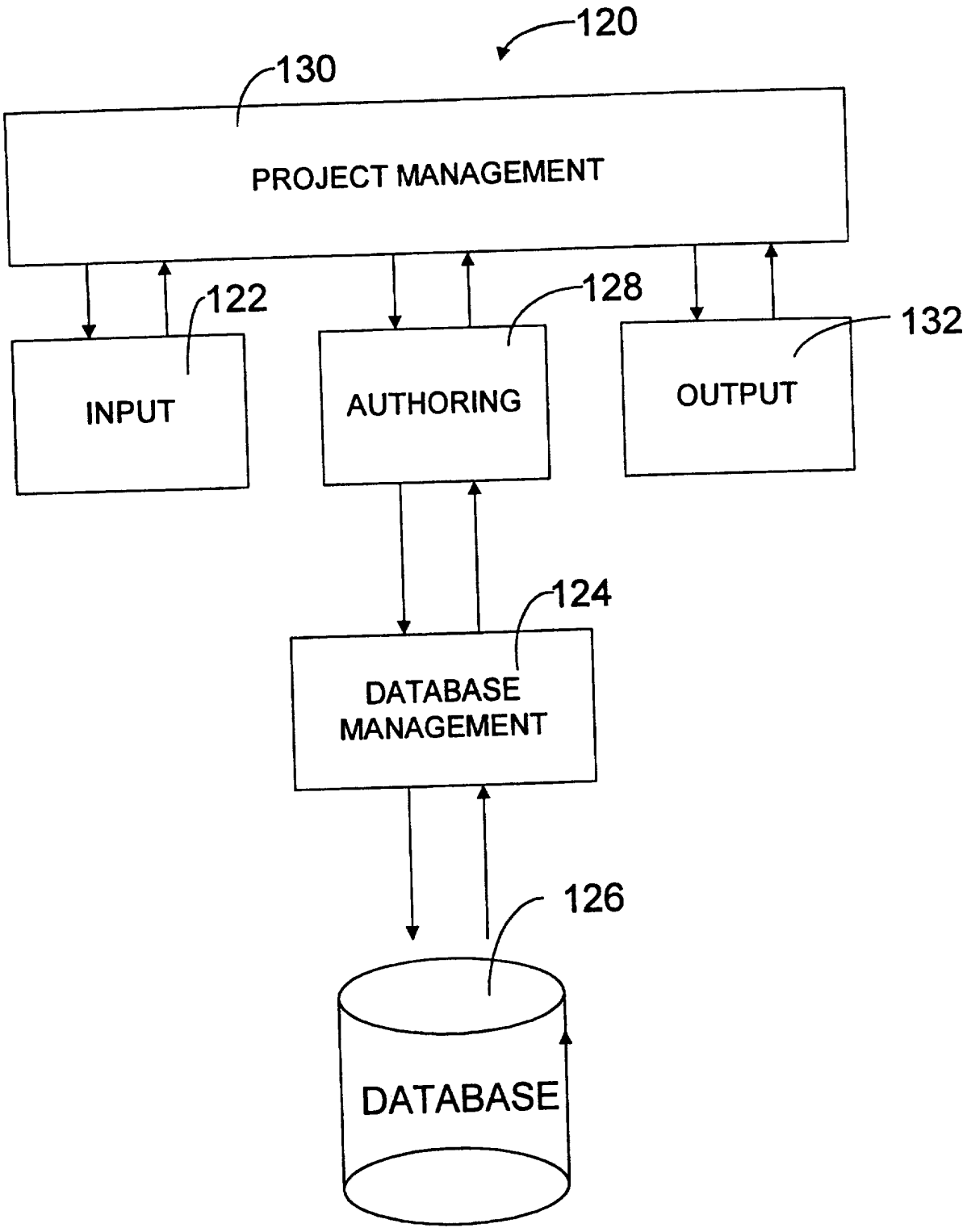

120
130
PROJECT MANAGEMENT
122
128
132
INPUT
AUTHORING
OUTPUT
124
DATABASE MANAGEMENT
126
DATABASE

FIG. 1

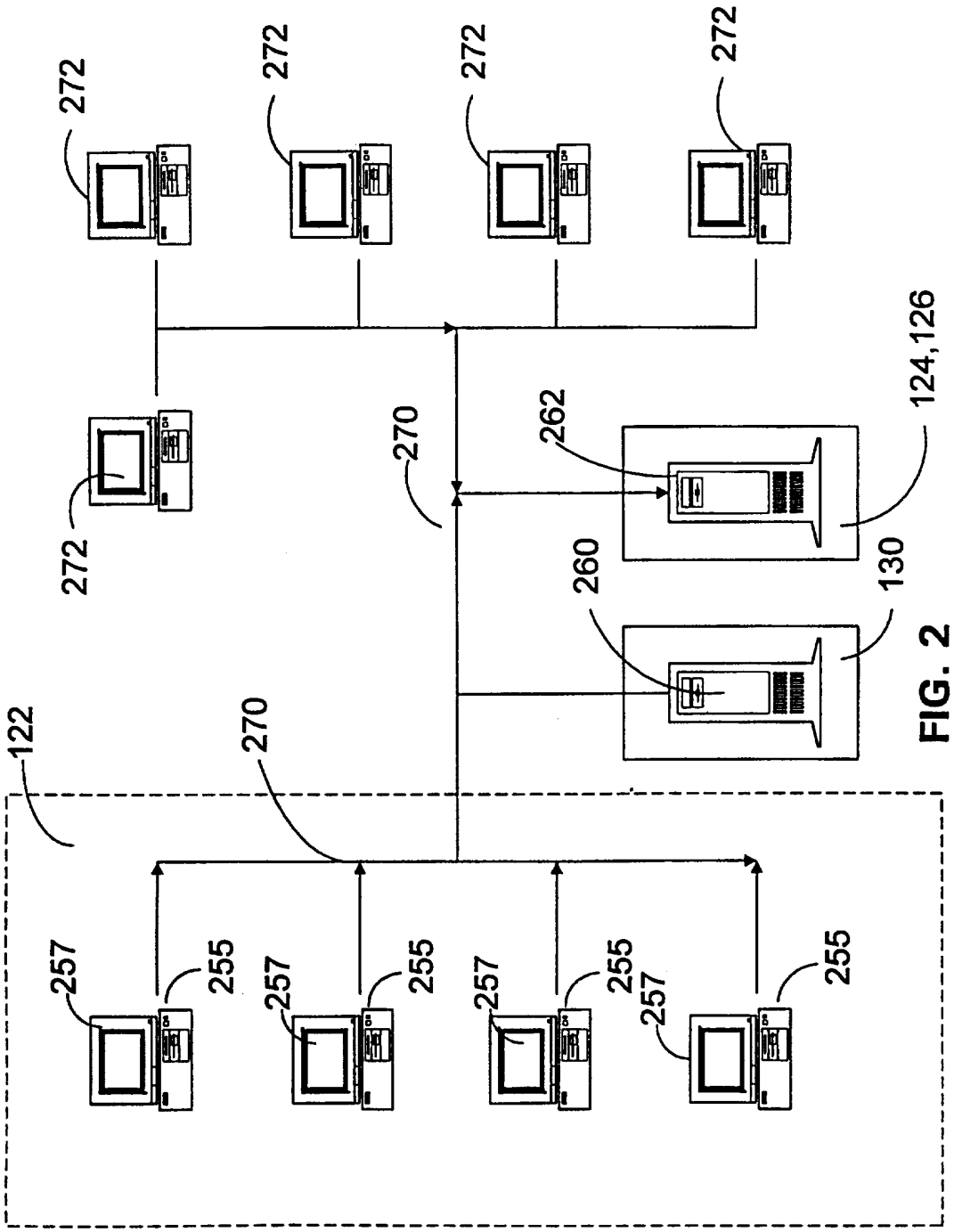

272
272
272
272
272
270
262
124,126
260
130
122
270
257
255
257
255
257
255
257
255

FIG. 2

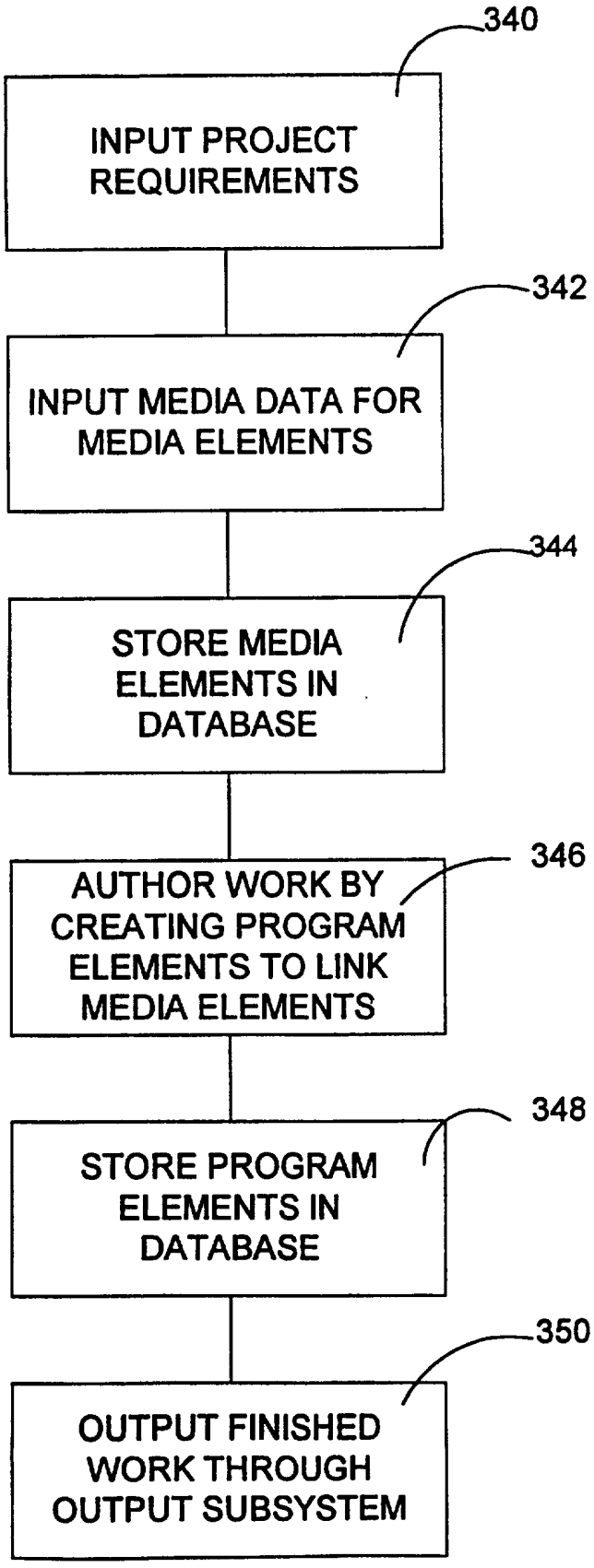
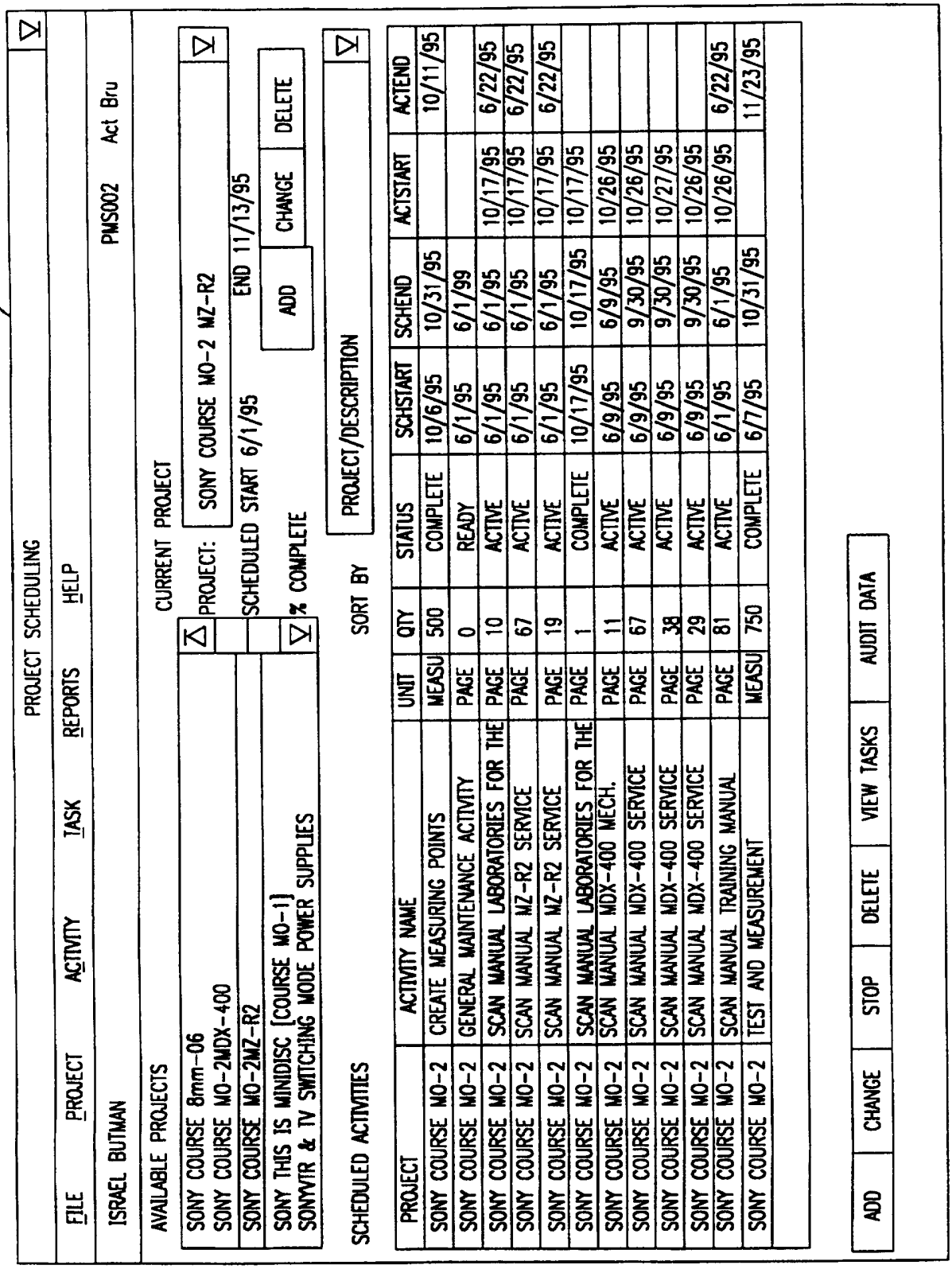

340
INPUT PROJECT REQUIREMENTS
342
INPUT MEDIA DATA FOR MEDIA ELEMENTS
344
STORE MEDIA ELEMENTS IN DATABASE
346
AUTHOR WORK BY CREATING PROGRAM ELEMENTS TO LINK MEDIA ELEMENTS
348
STORE PROGRAM ELEMENTS IN DATABASE
350
OUTPUT FINISHED WORK THROUGH OUTPUT SUBSYSTEM

FIG. 3



PROJECT	ACTIVITY NAME	UNIT	QTY	STATUS	SCHSTART	SCHEND	ACTSTART	ACTEND
SONY COURSE MO-2	CREATE MEASURING POINTS	MEASU	500	COMPLETE	10/6/95	10/31/95		10/11/95
SONY COURSE MO-2	GENERAL MAINTENANCE ACTIVITY	PAGE	0	READY	6/1/95	6/1/99		
SONY COURSE MO-2	SCAN MANUAL LABORATORIES FOR THE	PAGE	10	ACTIVE	6/1/95	6/1/95	10/17/95	6/22/95
SONY COURSE MO-2	SCAN MANUAL MZ-R2 SERVICE	PAGE	67	ACTIVE	6/1/95	6/1/95	10/17/95	6/22/95
SONY COURSE MO-2	SCAN MANUAL MZ-R2 SERVICE	PAGE	19	ACTIVE	6/1/95	6/1/95	10/17/95	6/22/95
SONY COURSE MO-2	SCAN MANUAL LABORATORIES FOR THE	PAGE	1	COMPLETE	10/17/95	10/17/95	10/17/95	
SONY COURSE MO-2	SCAN MANUAL MDX-400 MECH.	PAGE	11	ACTIVE	6/9/95	6/9/95	10/26/95	
SONY COURSE MO-2	SCAN MANUAL MDX-400 SERVICE	PAGE	67	ACTIVE	6/9/95	9/30/95	10/26/95	
SONY COURSE MO-2	SCAN MANUAL MDX-400 SERVICE	PAGE	38	ACTIVE	6/9/95	9/30/95	10/27/95	
SONY COURSE MO-2	SCAN MANUAL MDX-400 SERVICE	PAGE	29	ACTIVE	6/9/95	9/30/95	10/26/95	
SONY COURSE MO-2	SCAN MANUAL TRAINING MANUAL	PAGE	81	ACTIVE	6/1/95	6/1/95	10/26/95	6/22/95
SONY COURSE MO-2	TEST AND MEASUREMENT	MEASU	750	COMPLETE	6/7/95	10/31/95		11/23/95

FIG. 4

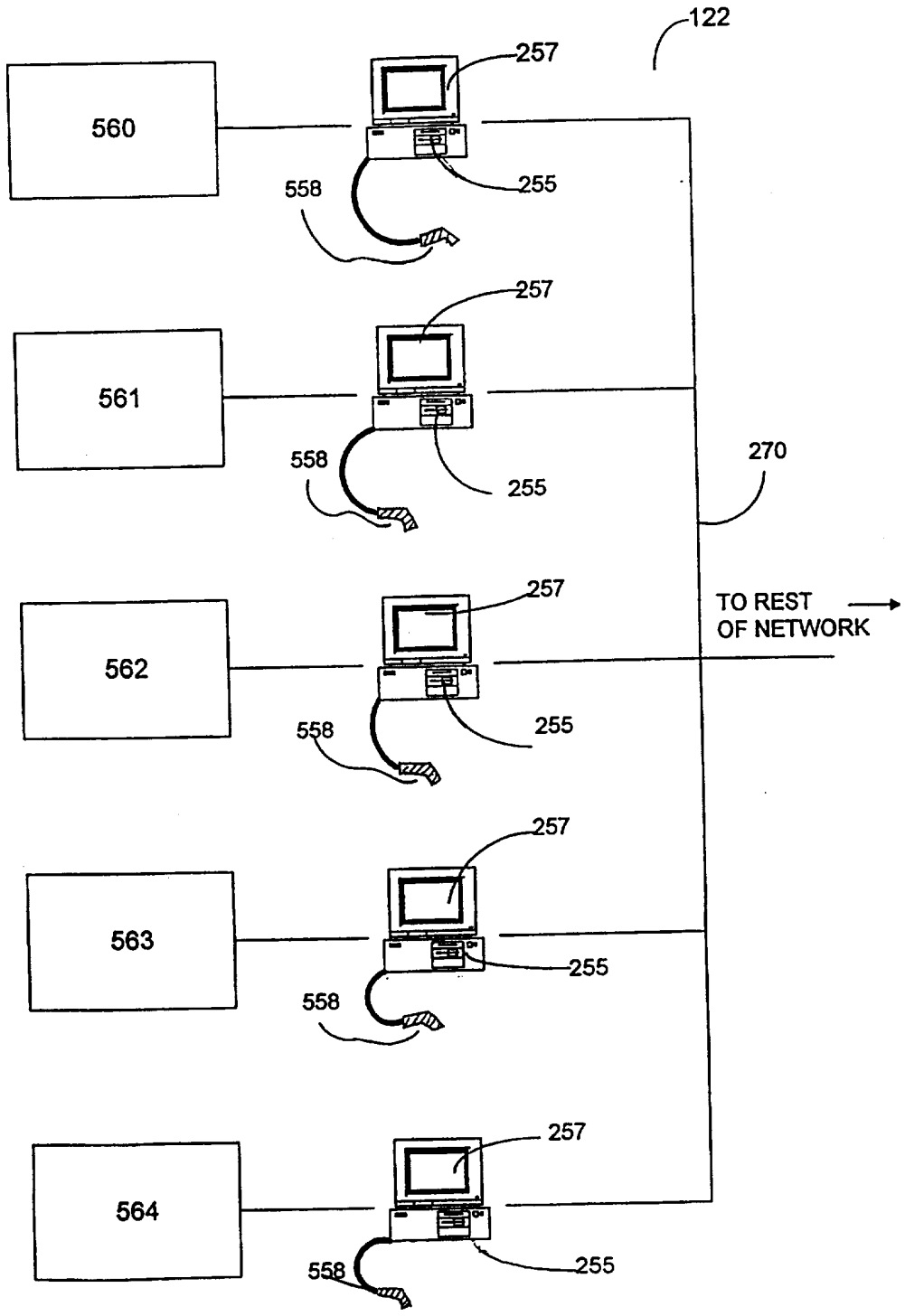

122
257
560
255
558
257
561
558
255
270
257
TO REST
OF NETWORK
562
558
255
257
563
255
558
257
564
558
255

FIG. 5

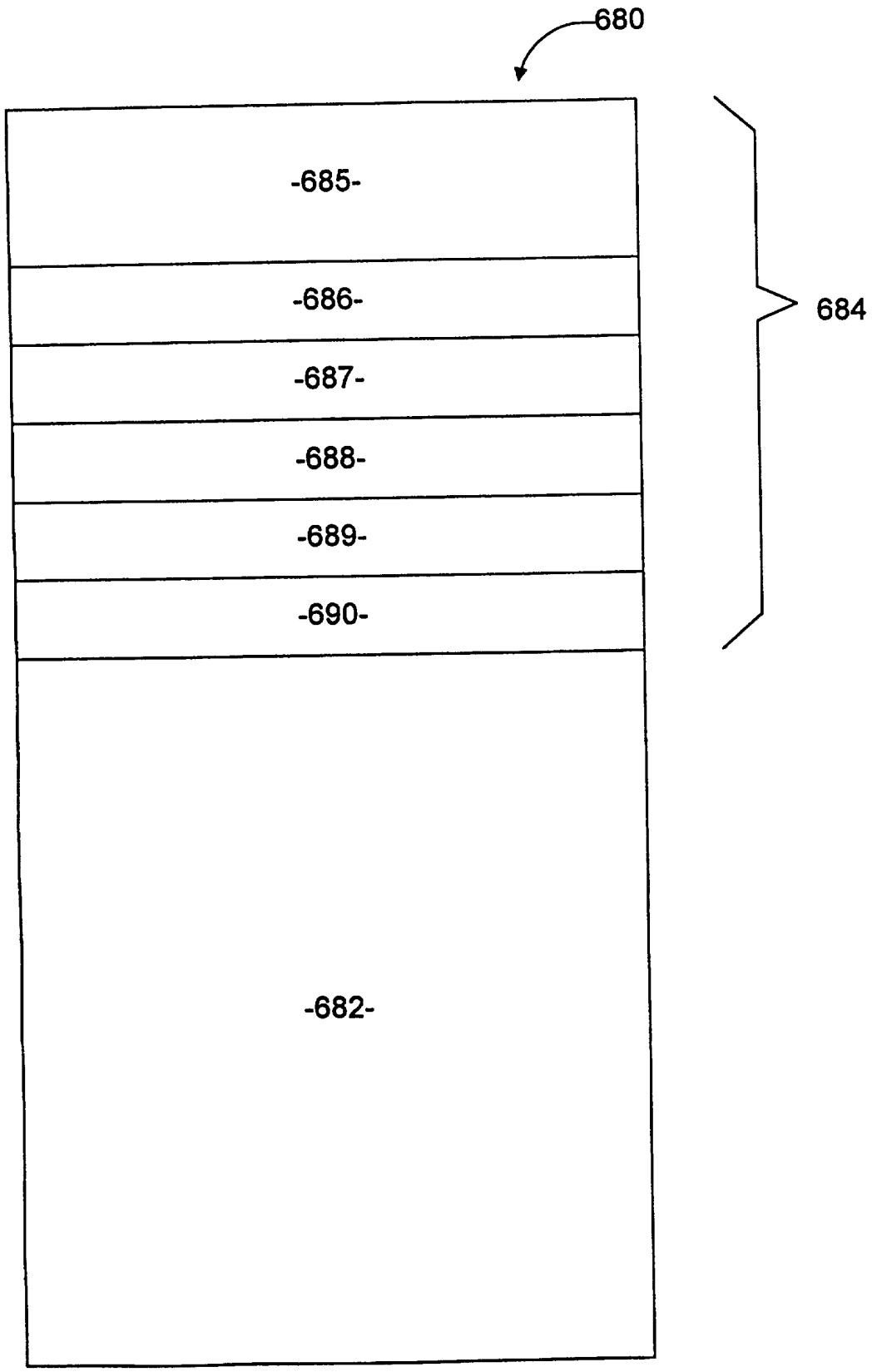


FIG. 6

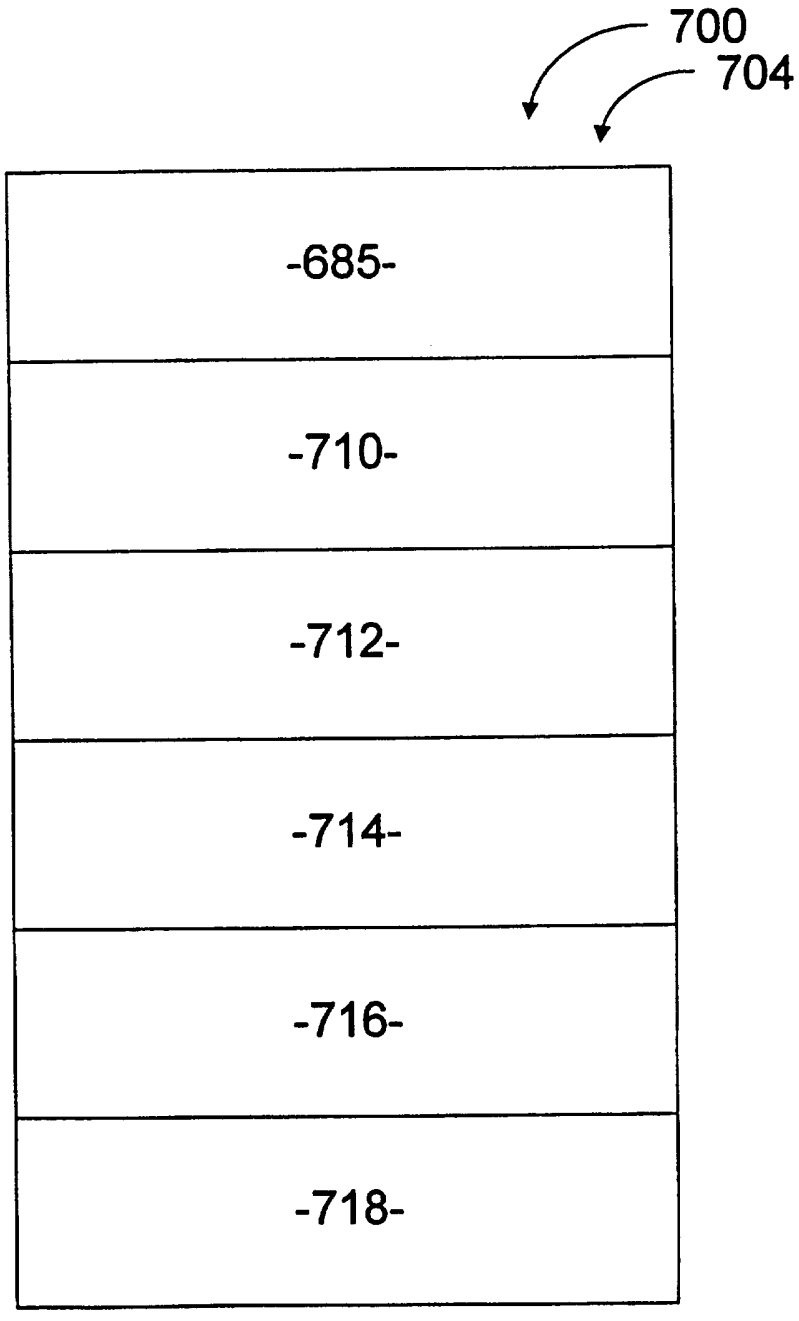


FIG. 7A

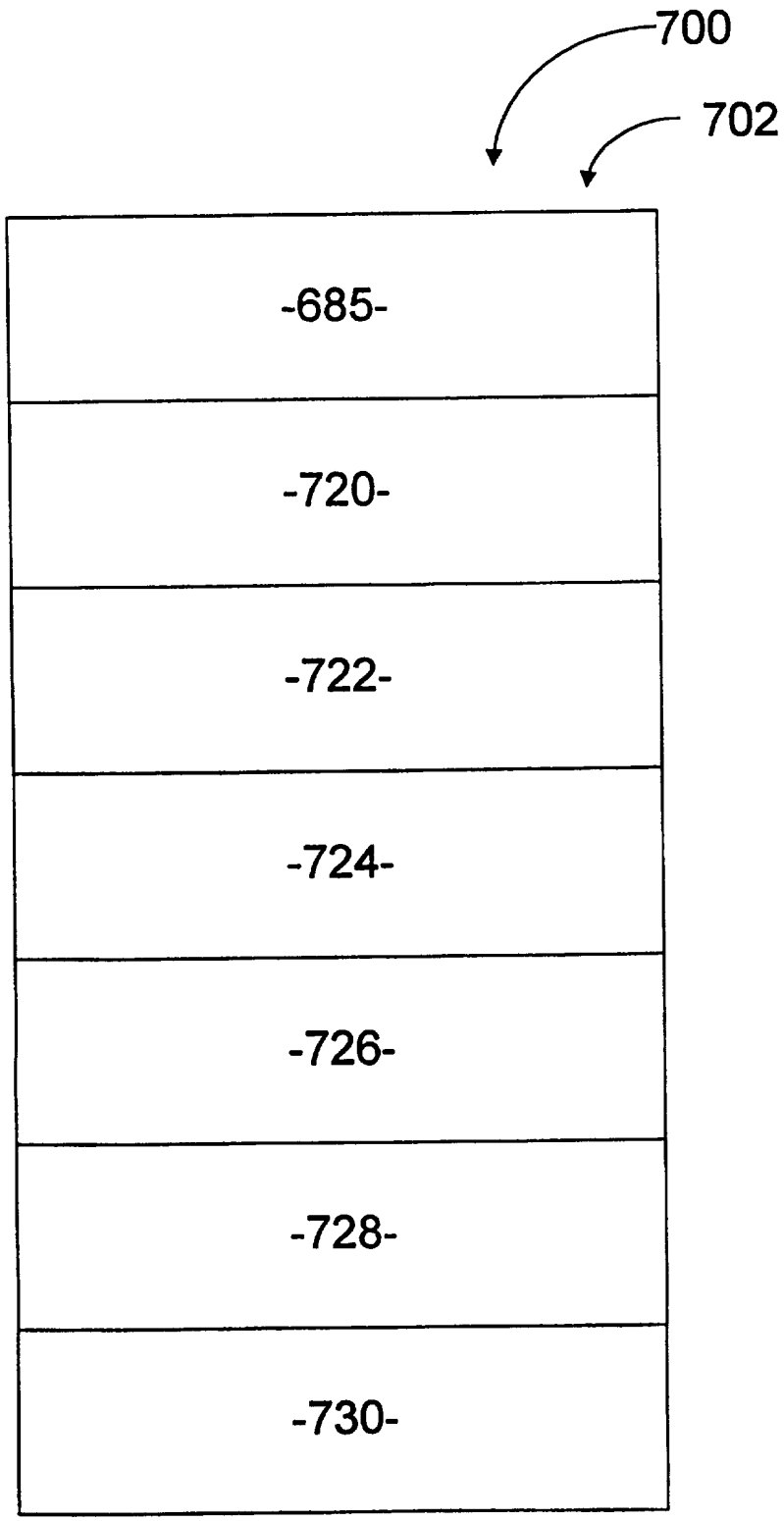


FIG. 7B

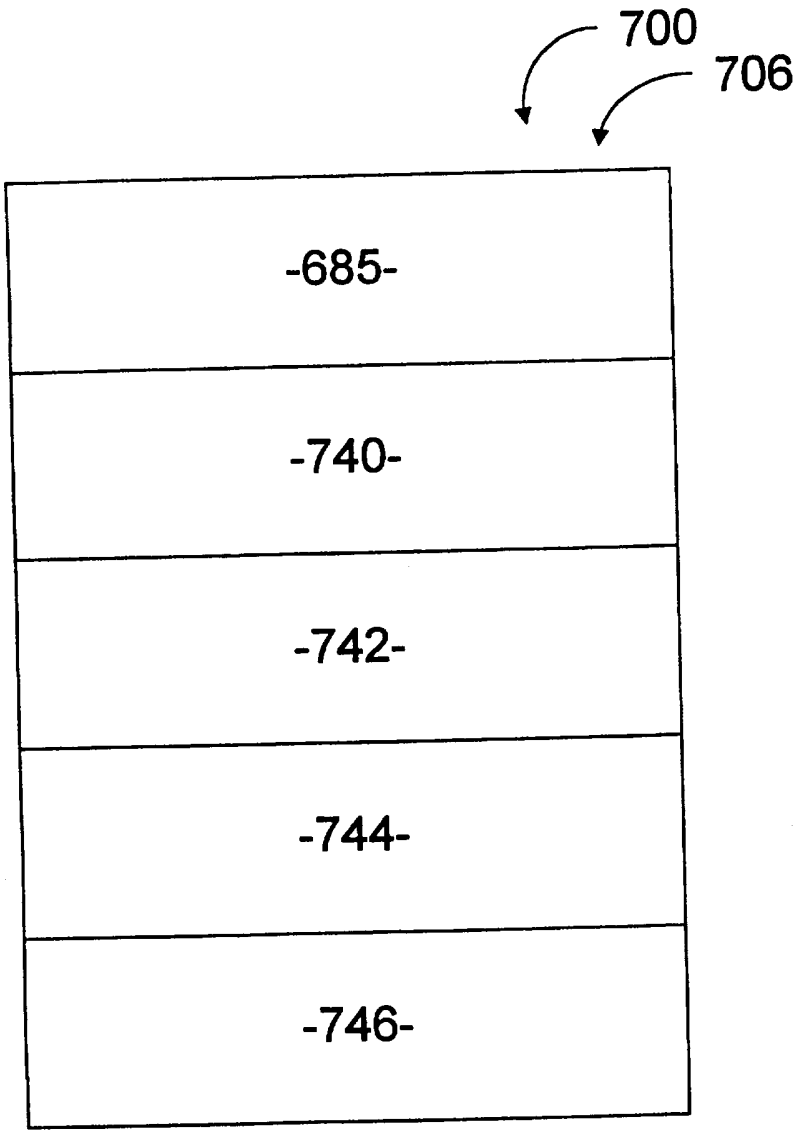


FIG. 7C

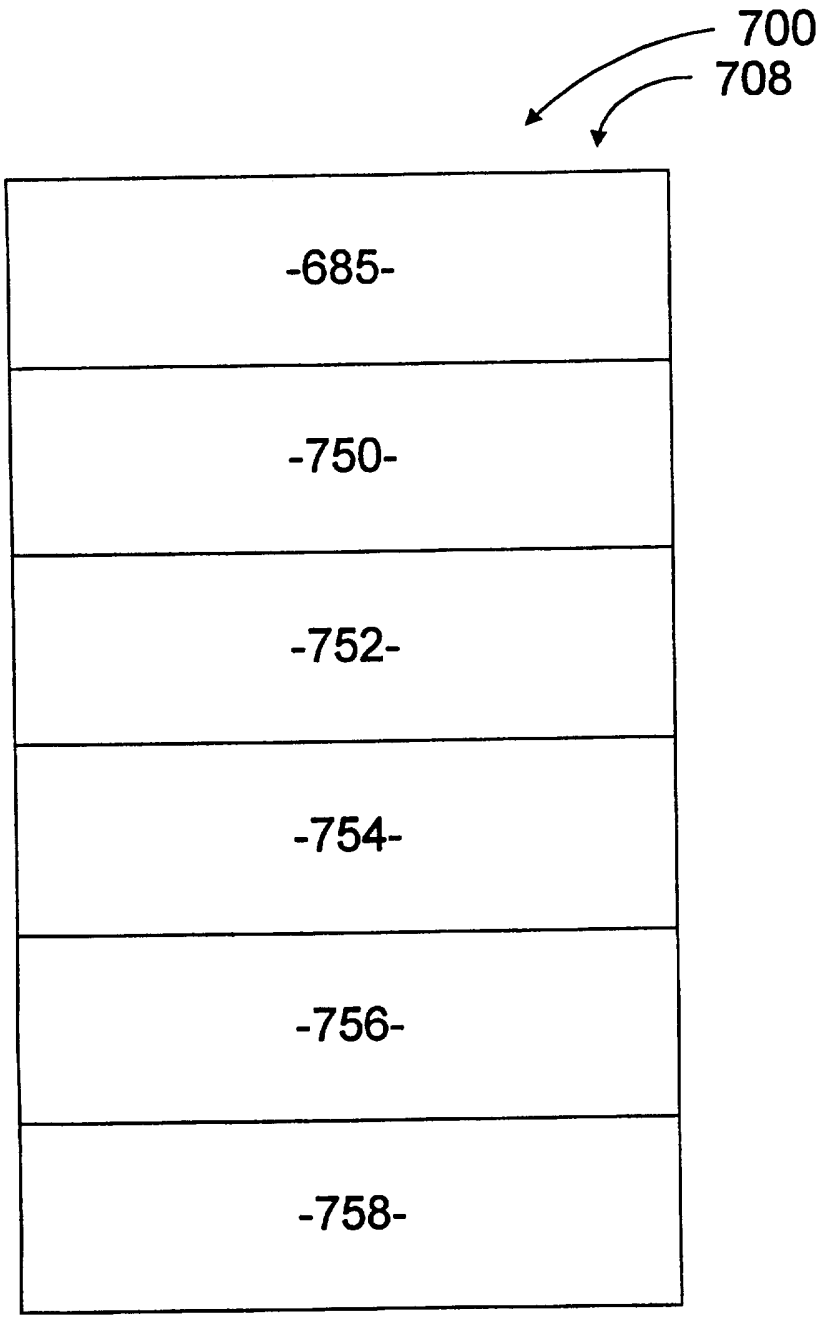


FIG. 7D

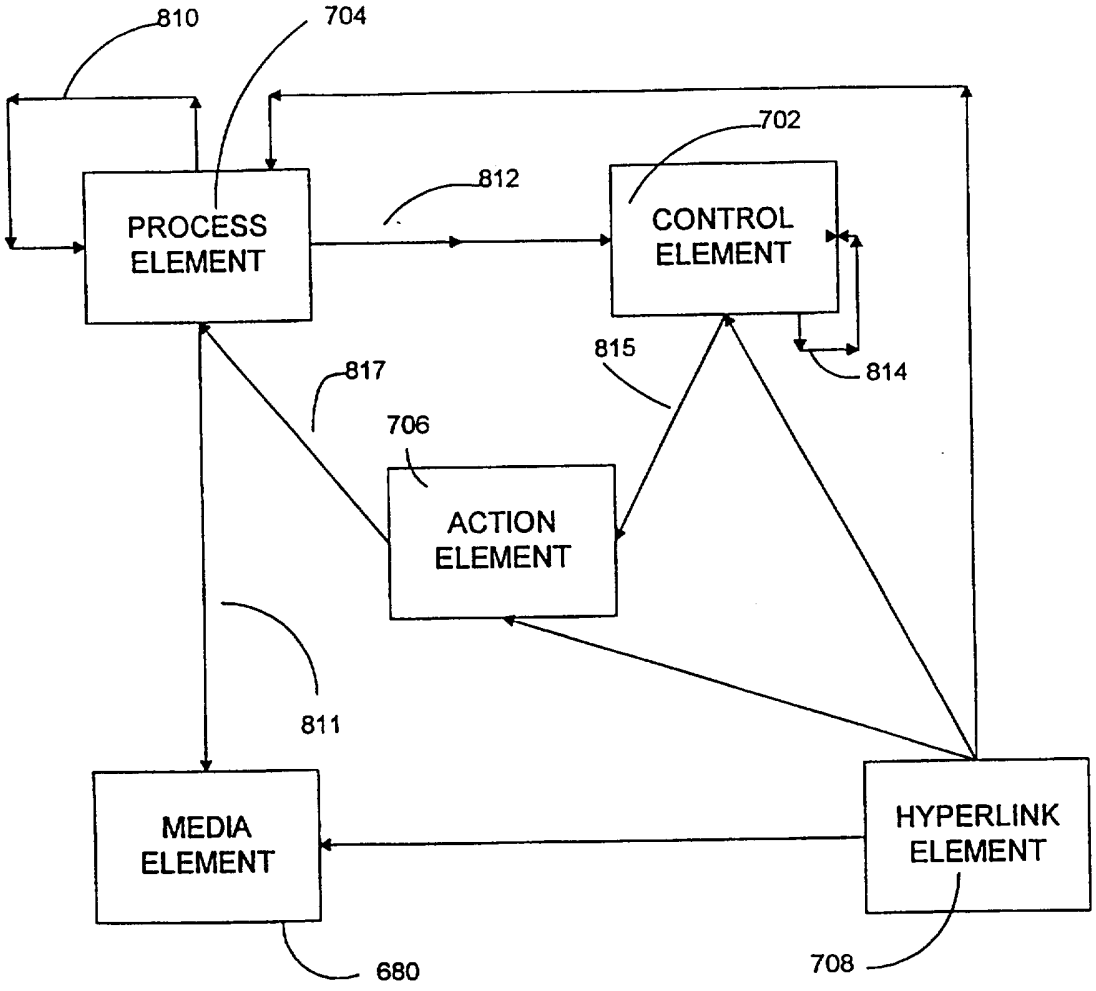

810
704
702
812
PROCESS ELEMENT
CONTROL ELEMENT
815
814
817
706
ACTION ELEMENT
811
MEDIA ELEMENT
HYPERLINK ELEMENT
680
708

FIG. 8

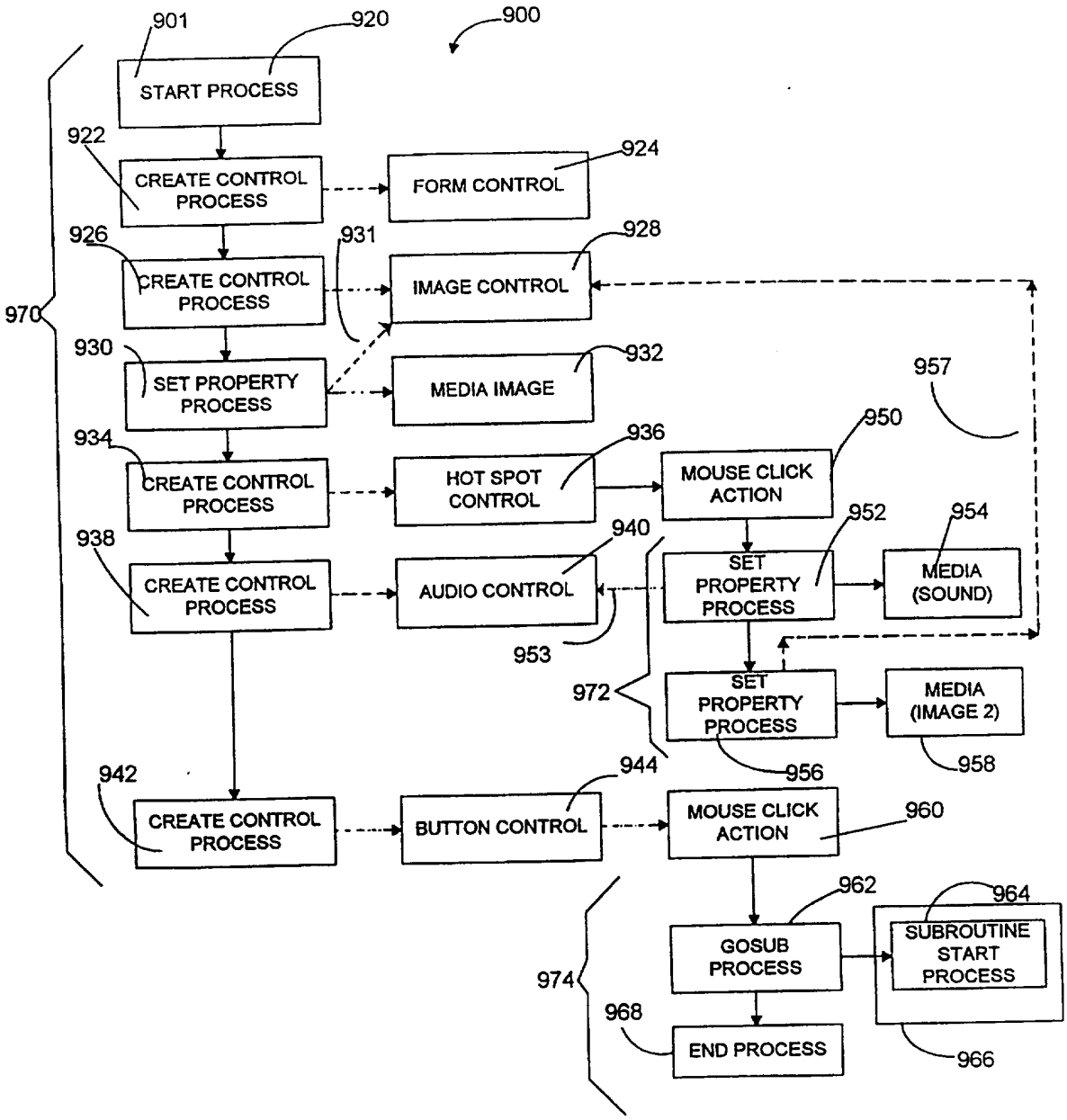

901
920
900
START PROCESS
922
CREATE CONTROL PROCESS
FORM CONTROL
924
926
CREATE CONTROL PROCESS
931
IMAGE CONTROL
928
970
930
SET PROPERTY PROCESS
MEDIA IMAGE
932
957
934
CREATE CONTROL PROCESS
HOT SPOT CONTROL
936
MOUSE CLICK ACTION
950
938
CREATE CONTROL PROCESS
AUDIO CONTROL
940
SET PROPERTY PROCESS
952
954
MEDIA (SOUND)
953
972
SET PROPERTY PROCESS
MEDIA (IMAGE 2)
956
958
942
CREATE CONTROL PROCESS
BUTTON CONTROL
944
MOUSE CLICK ACTION
960
962
964
GOSUB PROCESS
SUBROUTINE START PROCESS
974
968
END PROCESS
966

FIG. 9

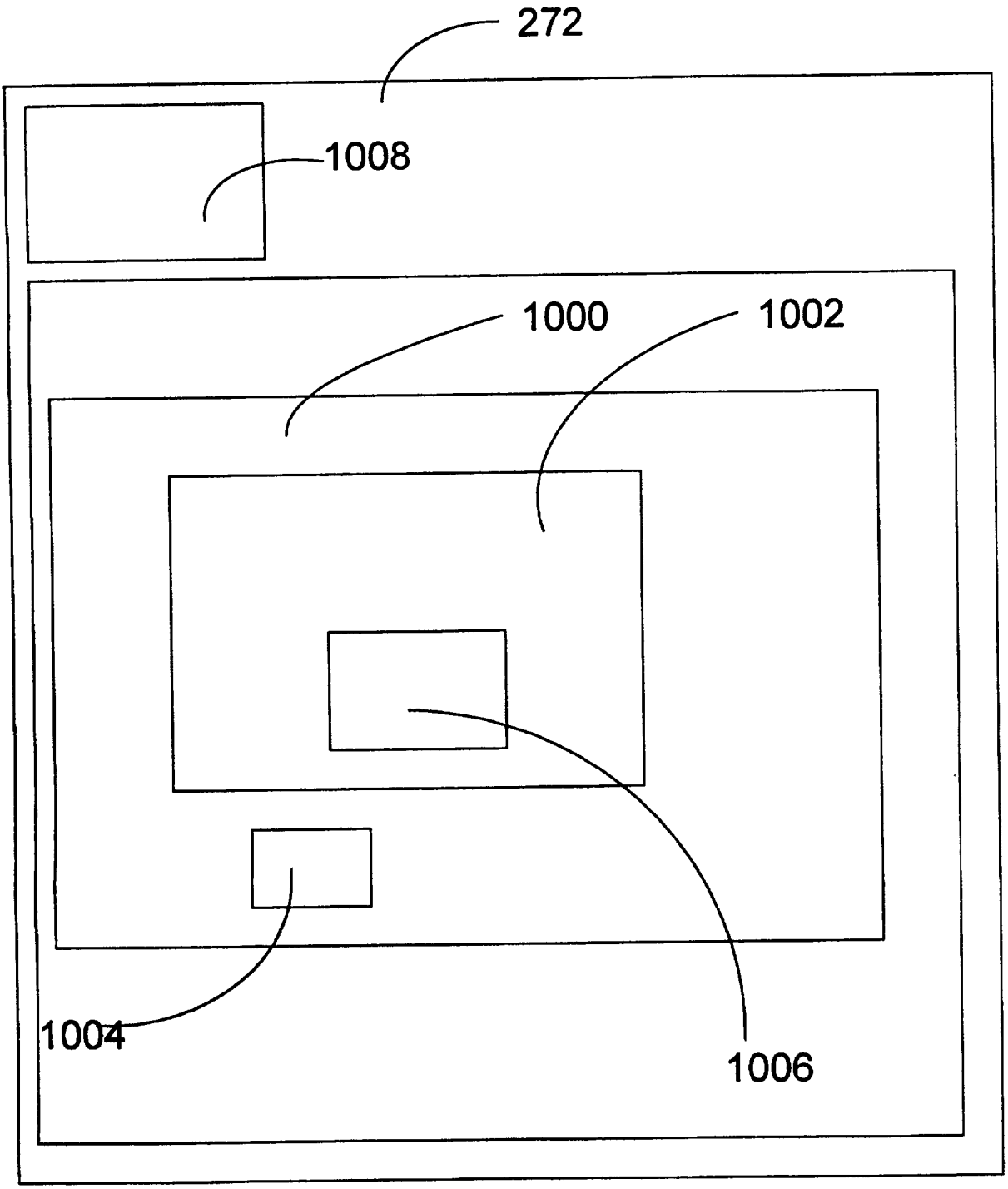


FIG. 10

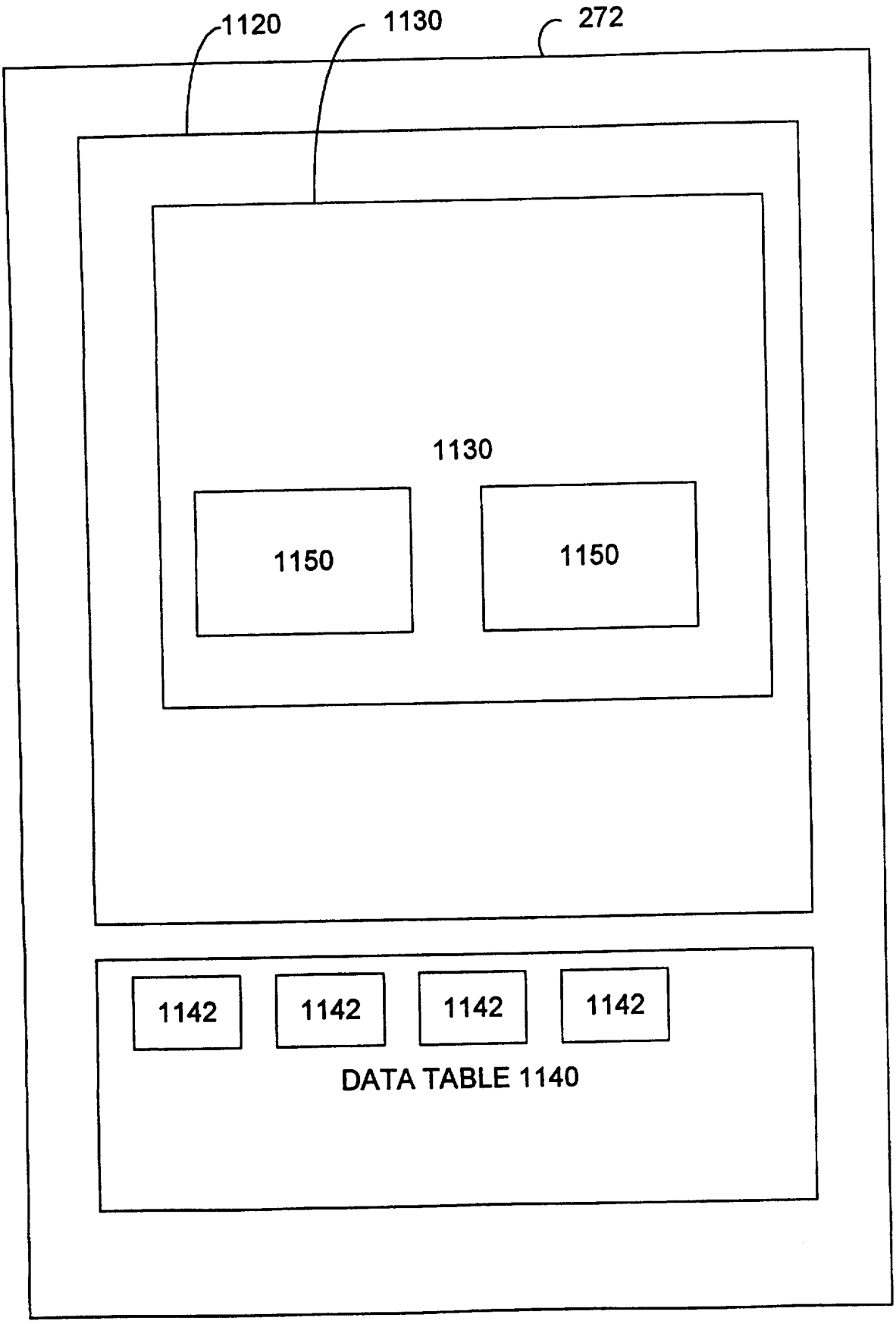

1120
1130
272
1130
1150
1150
1142
1142
1142
1142
DATA TABLE 1140

FIG. 11A

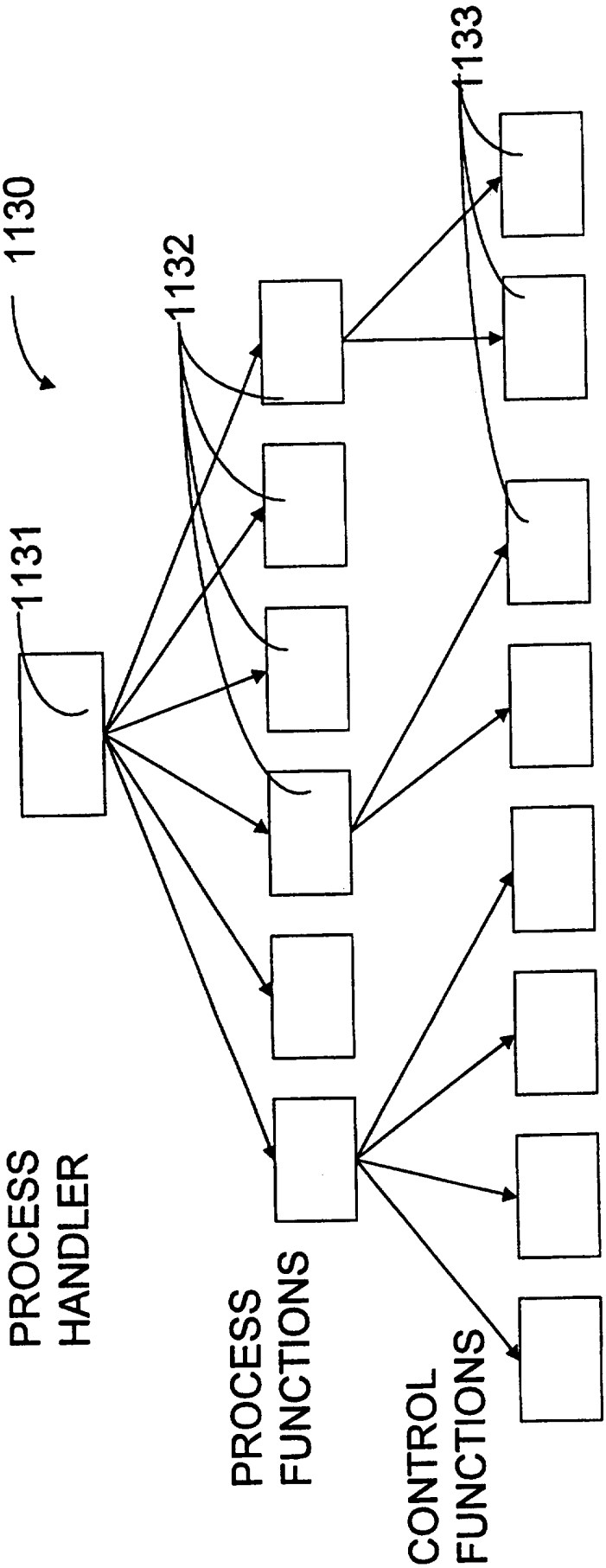

1130
1131
1132
1133
PROCESS HANDLER
PROCESS FUNCTIONS
CONTROL FUNCTIONS

FIG. 11B

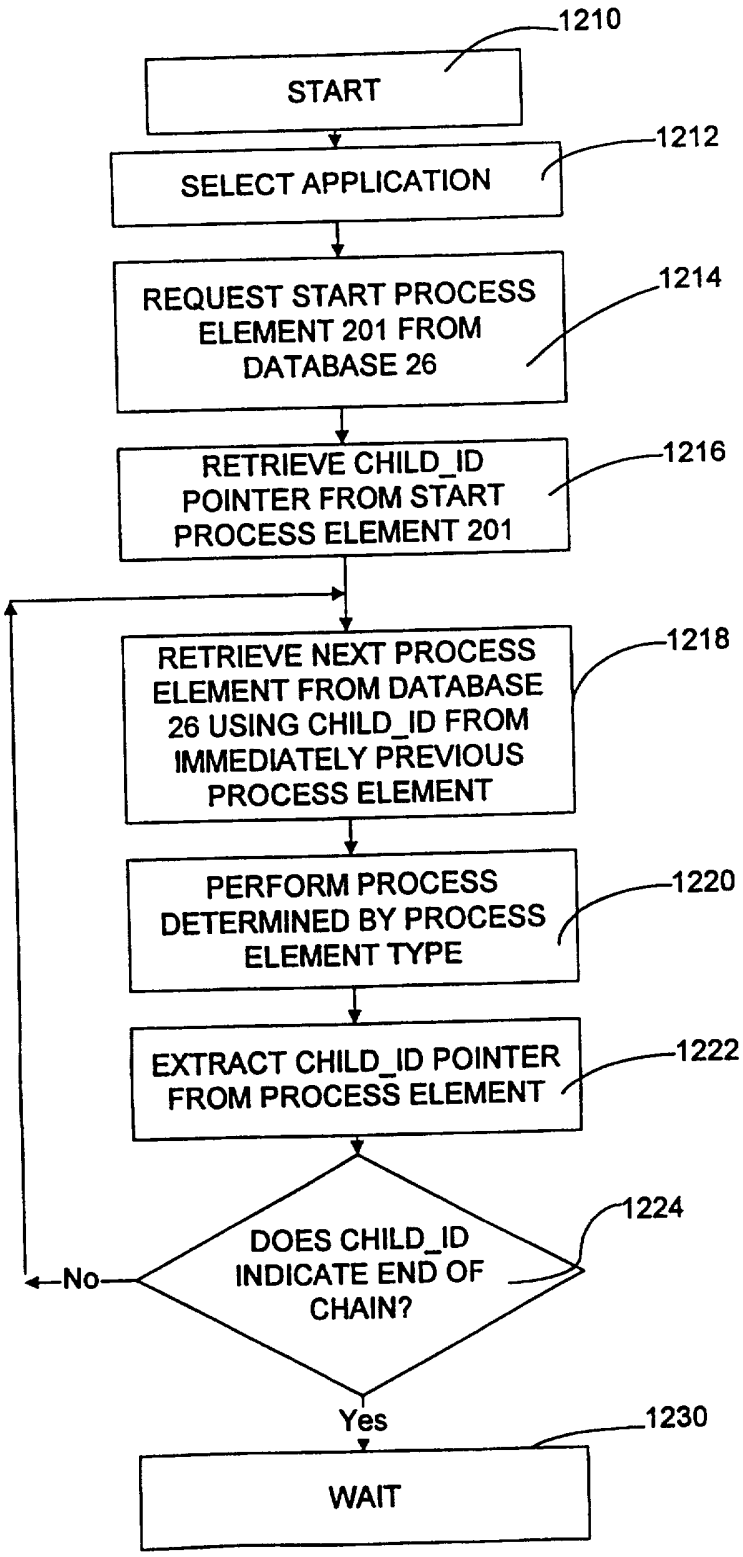

1210
START
1212
SELECT APPLICATION
1214
REQUEST START PROCESS ELEMENT 201 FROM DATABASE 26
1216
RETRIEVE CHILD_ID POINTER FROM START PROCESS ELEMENT 201
1218
RETRIEVE NEXT PROCESS ELEMENT FROM DATABASE 26 USING CHILD_ID FROM IMMEDIATELY PREVIOUS PROCESS ELEMENT
1220
PERFORM PROCESS DETERMINED BY PROCESS ELEMENT TYPE
1222
EXTRACT CHILD_ID POINTER FROM PROCESS ELEMENT
1224
DOES CHILD_ID INDICATE END OF CHAIN?
No
Yes
1230
WAIT

FIG. 12

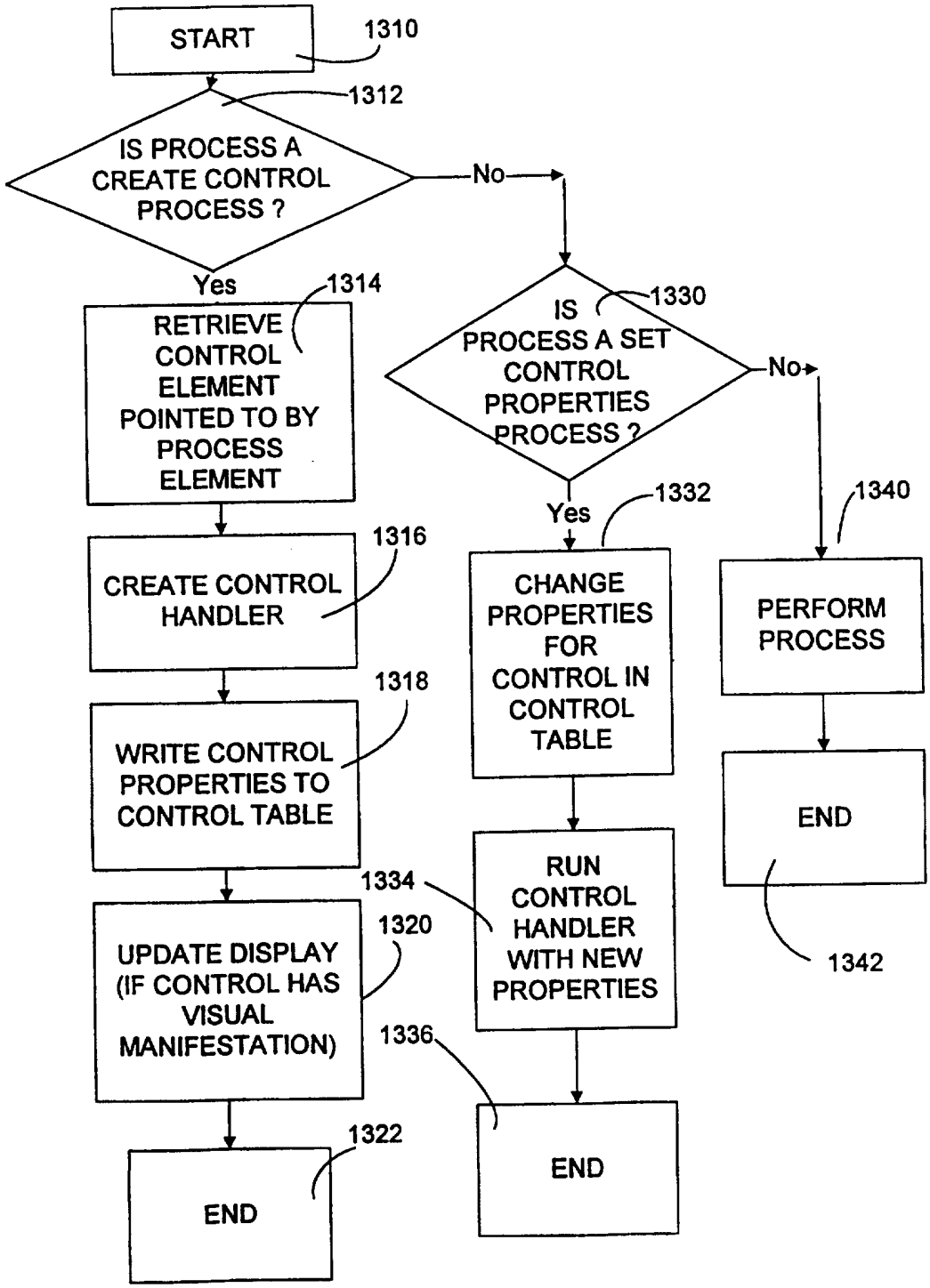

START
1310
1312
IS PROCESS A CREATE CONTROL PROCESS ?
No
Yes
1314
RETRIEVE CONTROL ELEMENT POINTED TO BY PROCESS ELEMENT
1316
CREATE CONTROL HANDLER
1318
WRITE CONTROL PROPERTIES TO CONTROL TABLE
1320
UPDATE DISPLAY (IF CONTROL HAS VISUAL MANIFESTATION)
1322
END
1330
IS PROCESS A SET CONTROL PROPERTIES PROCESS ?
No
Yes
1332
CHANGE PROPERTIES FOR CONTROL IN CONTROL TABLE
1334
RUN CONTROL HANDLER WITH NEW PROPERTIES
1336
END
1340
PERFORM PROCESS
END
1342

FIG. 13

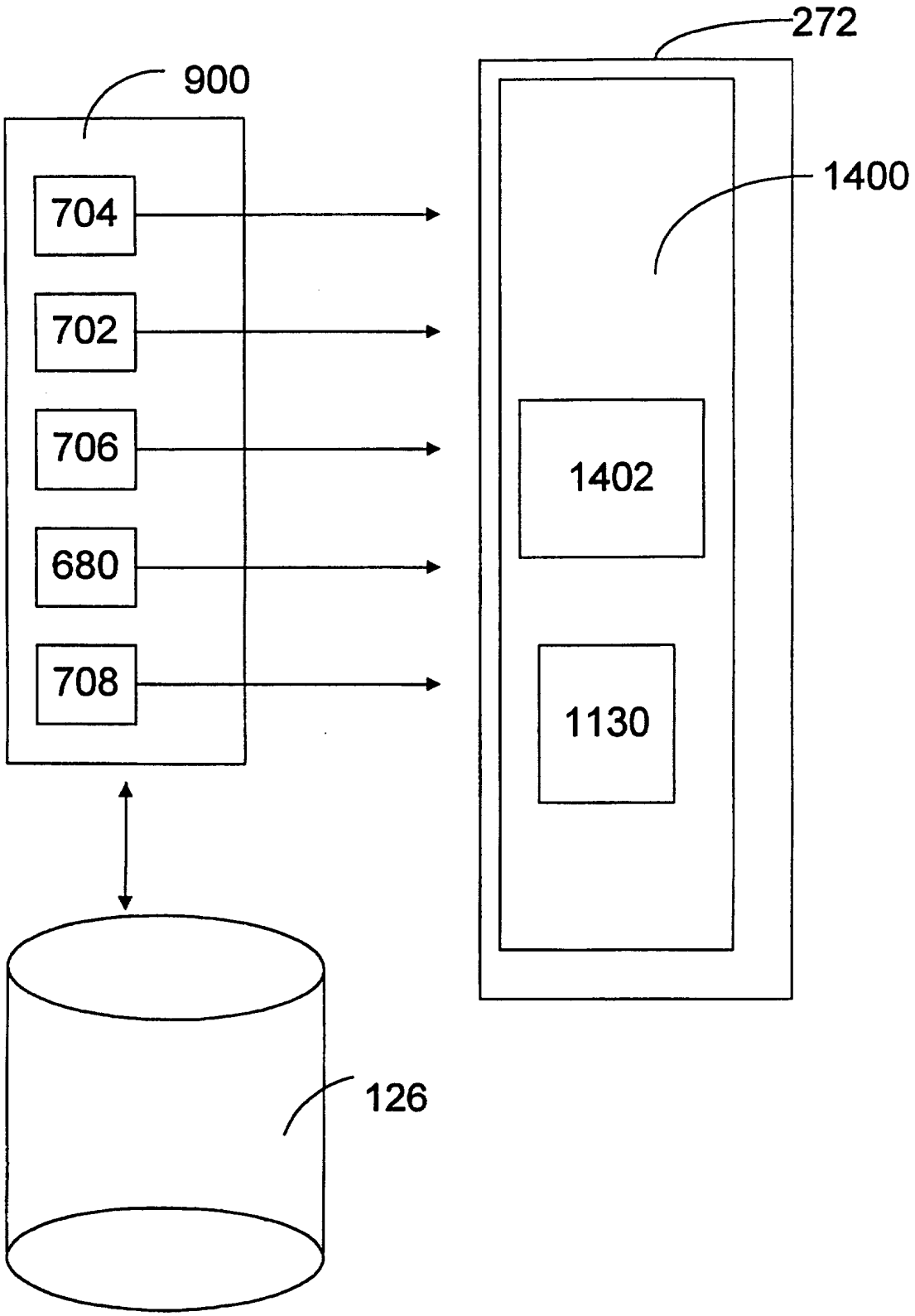


FIG. 14

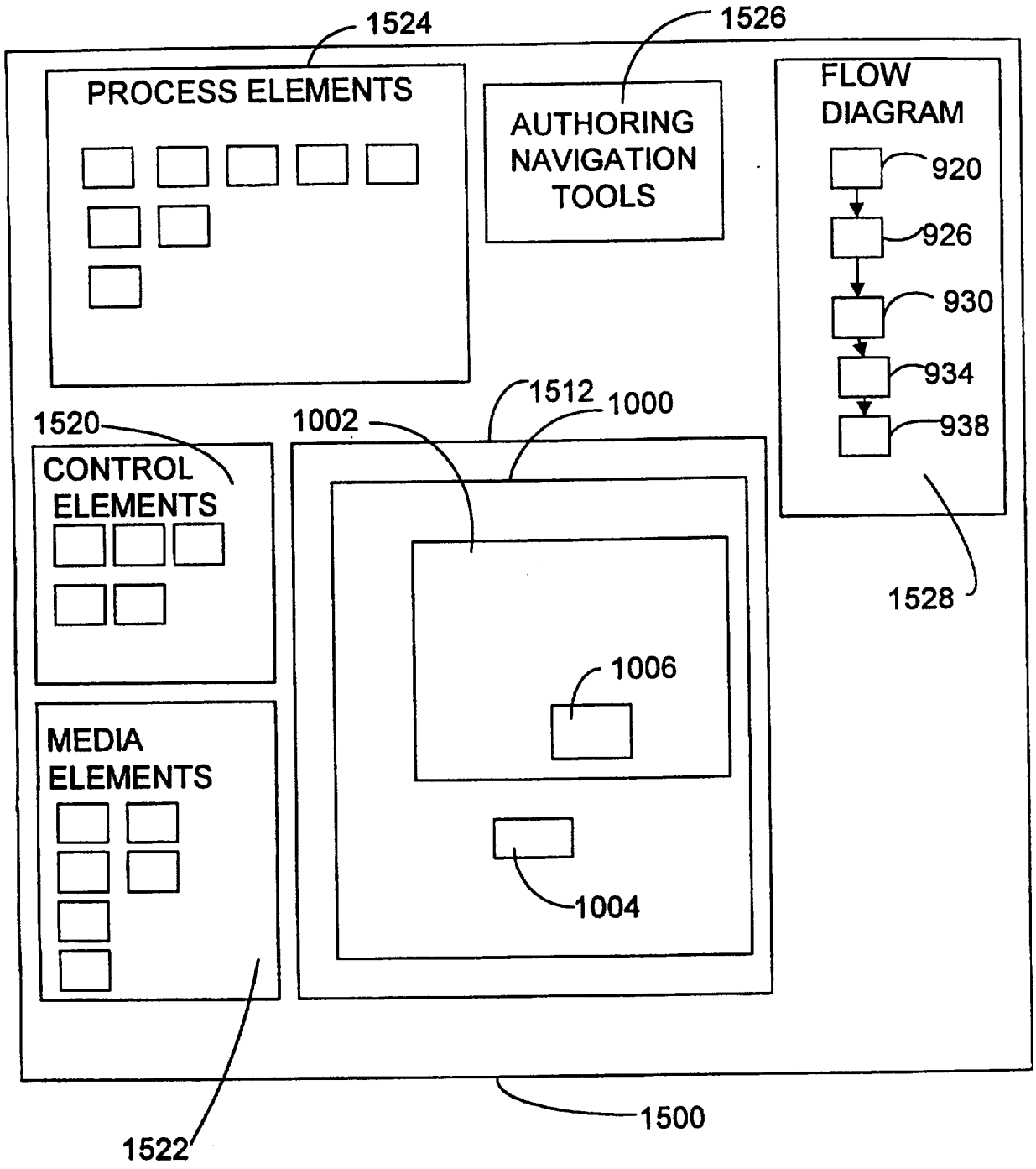

1524
1526
PROCESS ELEMENTS
AUTHORING NAVIGATION TOOLS
FLOW DIAGRAM
920
926
930
934
938
1528
1512
1000
1002
1520
CONTROL ELEMENTS
1006
MEDIA ELEMENTS
1004
1522
1500

FIG. 15

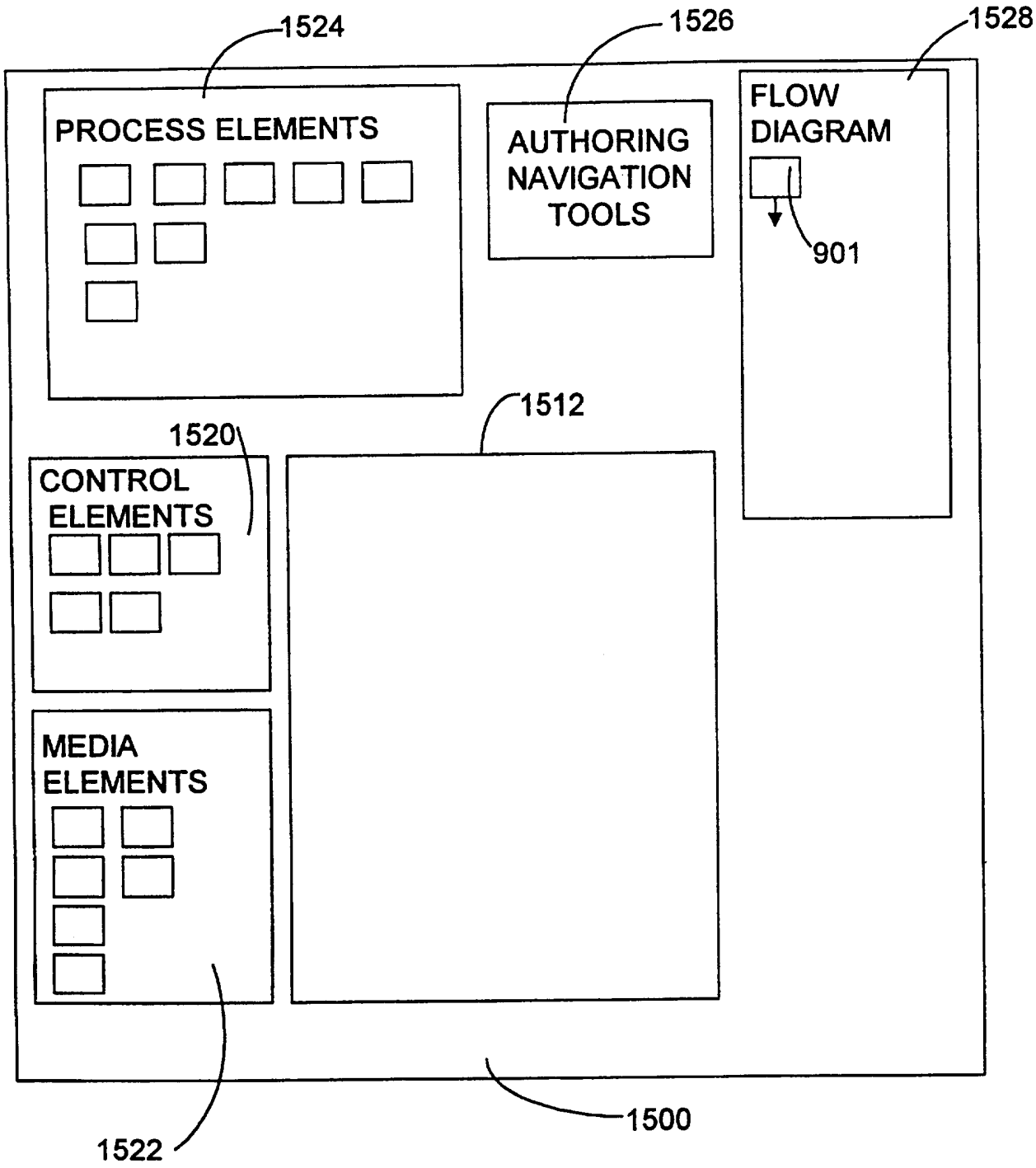

1524
1526
1528
PROCESS ELEMENTS
AUTHORING NAVIGATION TOOLS
FLOW DIAGRAM
901
1512
1520
CONTROL ELEMENTS
MEDIA ELEMENTS
1522
1500

FIG. 16

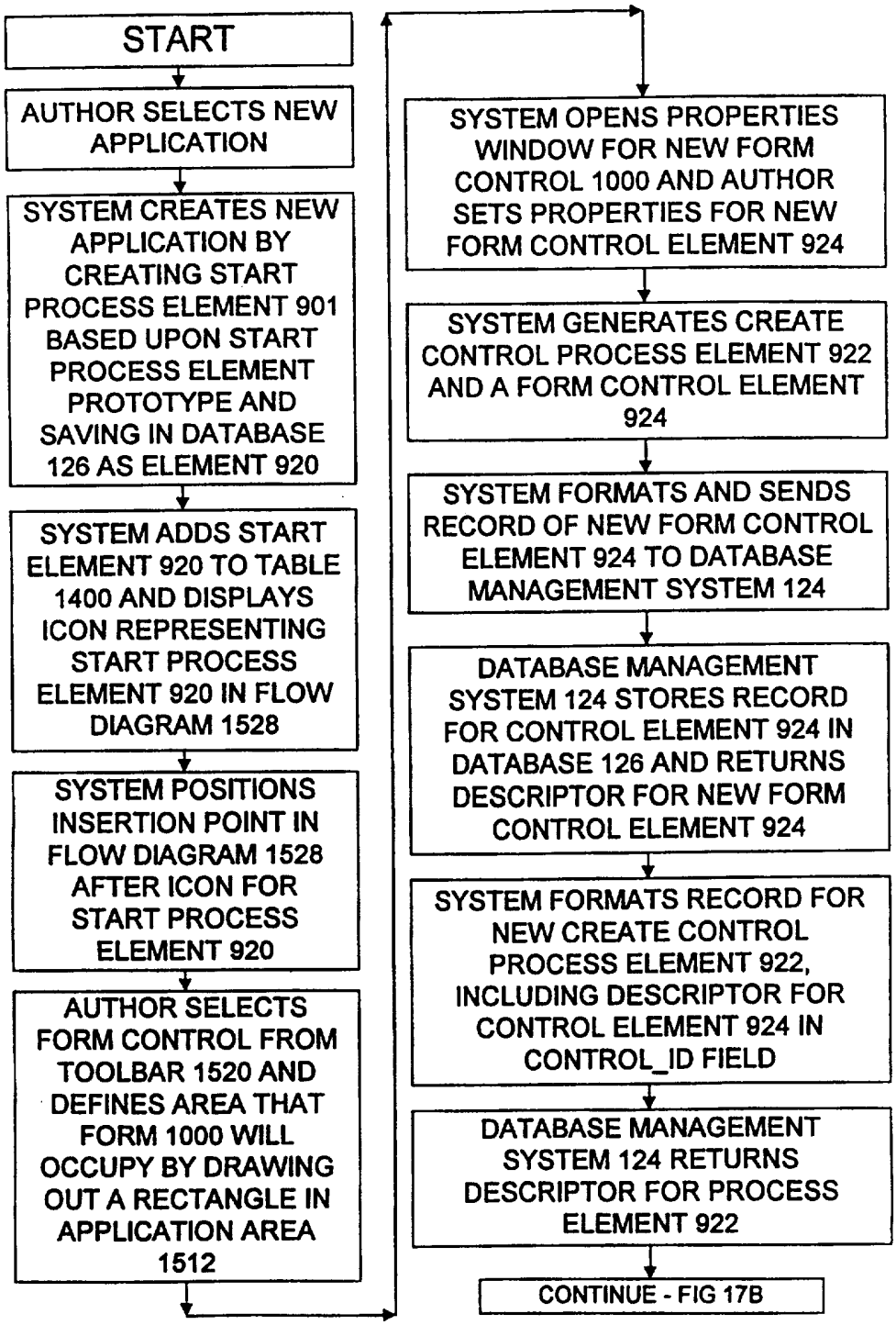

START
AUTHOR SELECTS NEW APPLICATION
SYSTEM CREATES NEW APPLICATION BY CREATING START PROCESS ELEMENT 901 BASED UPON START PROCESS ELEMENT PROTOTYPE AND SAVING IN DATABASE 126 AS ELEMENT 920
SYSTEM ADDS START ELEMENT 920 TO TABLE 1400 AND DISPLAYS ICON REPRESENTING START PROCESS ELEMENT 920 IN FLOW DIAGRAM 1528
SYSTEM POSITIONS INSERTION POINT IN FLOW DIAGRAM 1528 AFTER ICON FOR START PROCESS ELEMENT 920
AUTHOR SELECTS FORM CONTROL FROM TOOLBAR 1520 AND DEFINES AREA THAT FORM 1000 WILL OCCUPY BY DRAWING OUT A RECTANGLE IN APPLICATION AREA 1512
SYSTEM OPENS PROPERTIES WINDOW FOR NEW FORM CONTROL 1000 AND AUTHOR SETS PROPERTIES FOR NEW FORM CONTROL ELEMENT 924
SYSTEM GENERATES CREATE CONTROL PROCESS ELEMENT 922 AND A FORM CONTROL ELEMENT 924
SYSTEM FORMATS AND SENDS RECORD OF NEW FORM CONTROL ELEMENT 924 TO DATABASE MANAGEMENT SYSTEM 124
DATABASE MANAGEMENT SYSTEM 124 STORES RECORD FOR CONTROL ELEMENT 924 IN DATABASE 126 AND RETURNS DESCRIPTOR FOR NEW FORM CONTROL ELEMENT 924
SYSTEM FORMATS RECORD FOR NEW CREATE CONTROL PROCESS ELEMENT 922, INCLUDING DESCRIPTOR FOR CONTROL ELEMENT 924 IN CONTROL_ID FIELD
DATABASE MANAGEMENT SYSTEM 124 RETURNS DESCRIPTOR FOR PROCESS ELEMENT 922
CONTINUE - FIG 17B

FIG. 17A

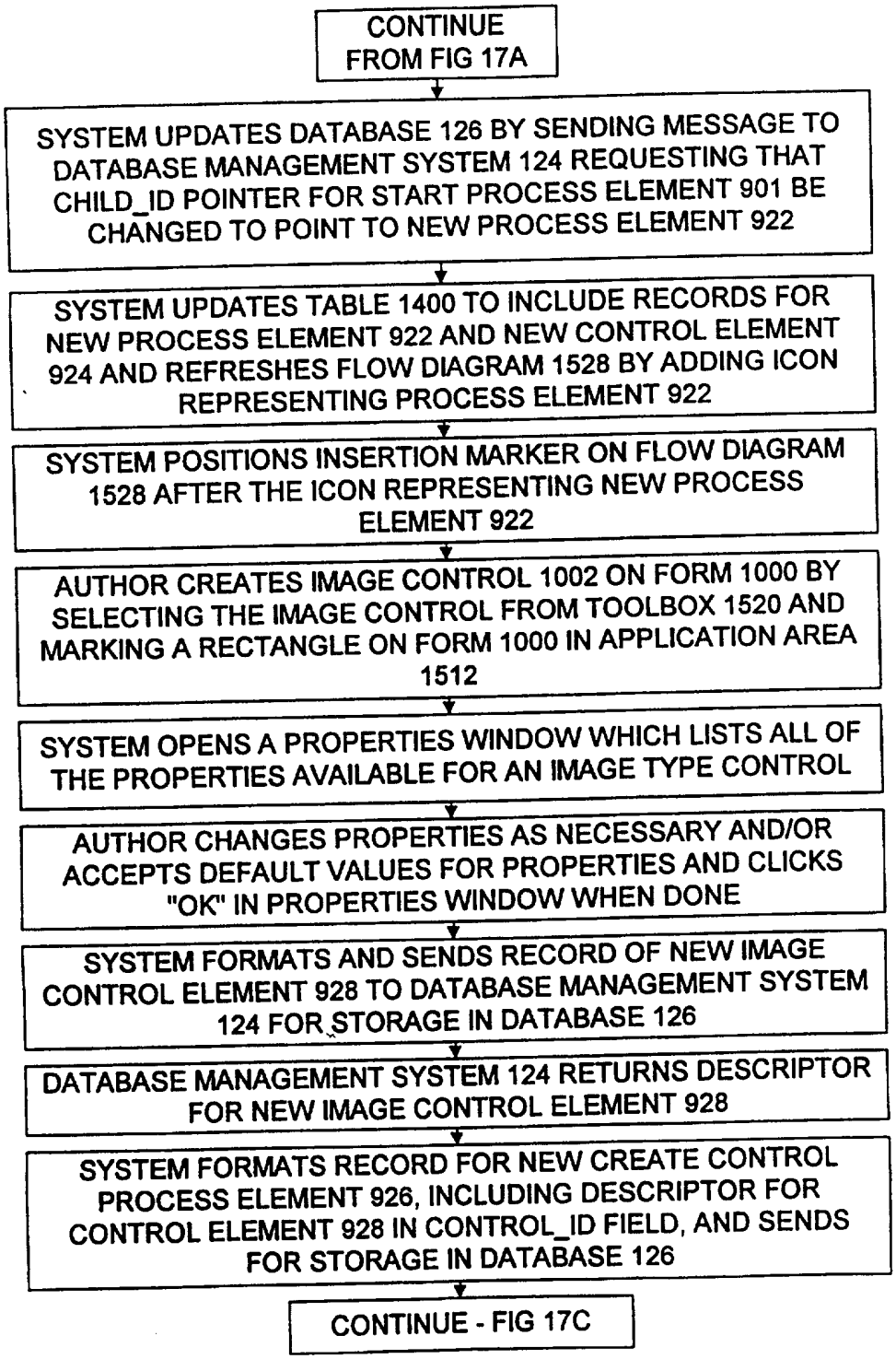

CONTINUE FROM FIG 17A
SYSTEM UPDATES DATABASE 126 BY SENDING MESSAGE TO DATABASE MANAGEMENT SYSTEM 124 REQUESTING THAT CHILD_ID POINTER FOR START PROCESS ELEMENT 901 BE CHANGED TO POINT TO NEW PROCESS ELEMENT 922
SYSTEM UPDATES TABLE 1400 TO INCLUDE RECORDS FOR NEW PROCESS ELEMENT 922 AND NEW CONTROL ELEMENT 924 AND REFRESHES FLOW DIAGRAM 1528 BY ADDING ICON REPRESENTING PROCESS ELEMENT 922
SYSTEM POSITIONS INSERTION MARKER ON FLOW DIAGRAM 1528 AFTER THE ICON REPRESENTING NEW PROCESS ELEMENT 922
AUTHOR CREATES IMAGE CONTROL 1002 ON FORM 1000 BY SELECTING THE IMAGE CONTROL FROM TOOLBOX 1520 AND MARKING A RECTANGLE ON FORM 1000 IN APPLICATION AREA 1512
SYSTEM OPENS A PROPERTIES WINDOW WHICH LISTS ALL OF THE PROPERTIES AVAILABLE FOR AN IMAGE TYPE CONTROL
AUTHOR CHANGES PROPERTIES AS NECESSARY AND/OR ACCEPTS DEFAULT VALUES FOR PROPERTIES AND CLICKS "OK" IN PROPERTIES WINDOW WHEN DONE
SYSTEM FORMATS AND SENDS RECORD OF NEW IMAGE CONTROL ELEMENT 928 TO DATABASE MANAGEMENT SYSTEM 124 FOR STORAGE IN DATABASE 126
DATABASE MANAGEMENT SYSTEM 124 RETURNS DESCRIPTOR FOR NEW IMAGE CONTROL ELEMENT 928
SYSTEM FORMATS RECORD FOR NEW CREATE CONTROL PROCESS ELEMENT 926, INCLUDING DESCRIPTOR FOR CONTROL ELEMENT 928 IN CONTROL_ID FIELD, AND SENDS FOR STORAGE IN DATABASE 126
CONTINUE - FIG 17C

FIG. 17B

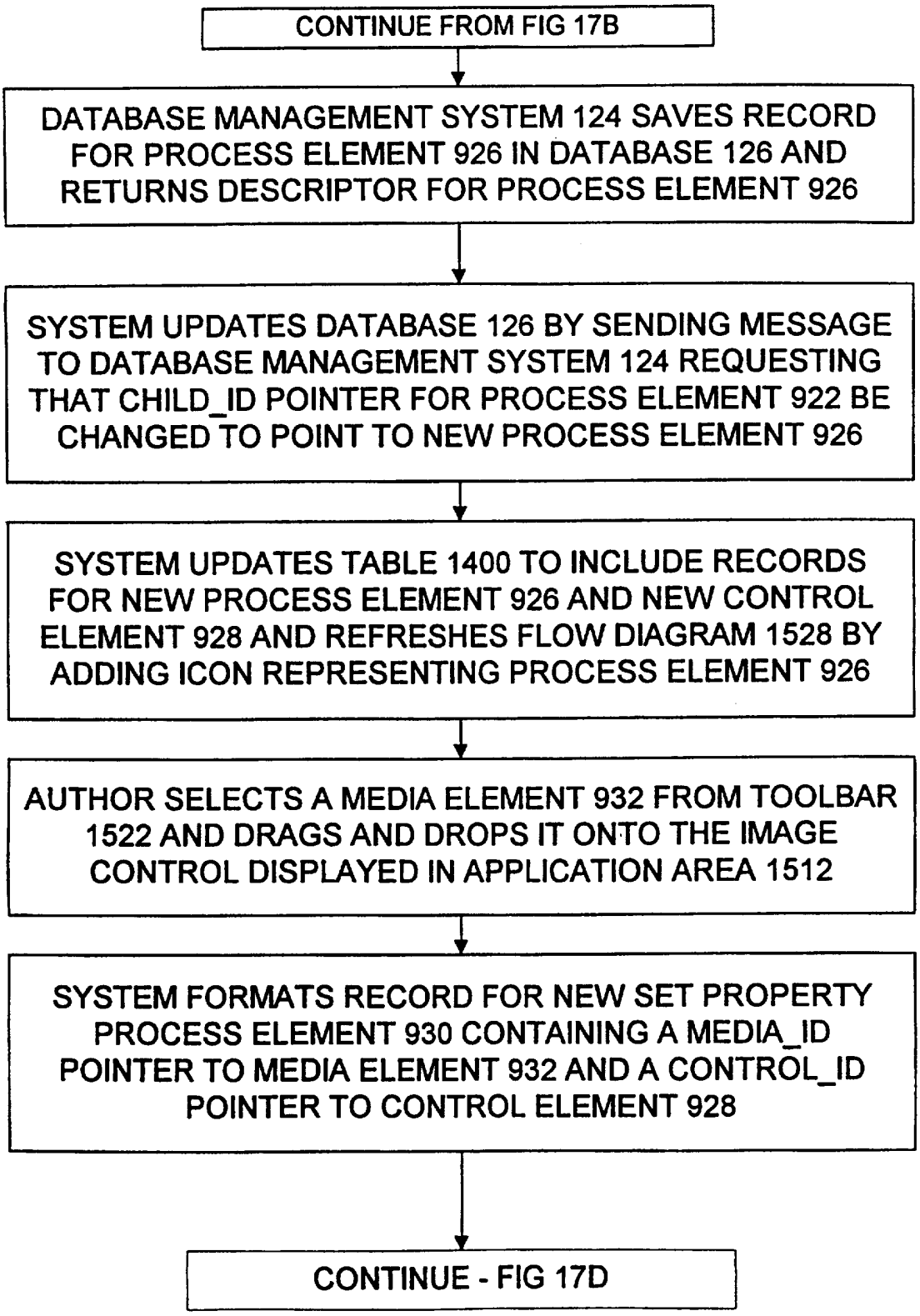

CONTINUE FROM FIG 17B
DATABASE MANAGEMENT SYSTEM 124 SAVES RECORD FOR PROCESS ELEMENT 926 IN DATABASE 126 AND RETURNS DESCRIPTOR FOR PROCESS ELEMENT 926
SYSTEM UPDATES DATABASE 126 BY SENDING MESSAGE TO DATABASE MANAGEMENT SYSTEM 124 REQUESTING THAT CHILD_ID POINTER FOR PROCESS ELEMENT 922 BE CHANGED TO POINT TO NEW PROCESS ELEMENT 926
SYSTEM UPDATES TABLE 1400 TO INCLUDE RECORDS FOR NEW PROCESS ELEMENT 926 AND NEW CONTROL ELEMENT 928 AND REFRESHES FLOW DIAGRAM 1528 BY ADDING ICON REPRESENTING PROCESS ELEMENT 926
AUTHOR SELECTS A MEDIA ELEMENT 932 FROM TOOLBAR 1522 AND DRAGS AND DROPS IT ONTO THE IMAGE CONTROL DISPLAYED IN APPLICATION AREA 1512
SYSTEM FORMATS RECORD FOR NEW SET PROPERTY PROCESS ELEMENT 930 CONTAINING A MEDIA_ID POINTER TO MEDIA ELEMENT 932 AND A CONTROL_ID POINTER TO CONTROL ELEMENT 928
CONTINUE - FIG 17D

FIG. 17C

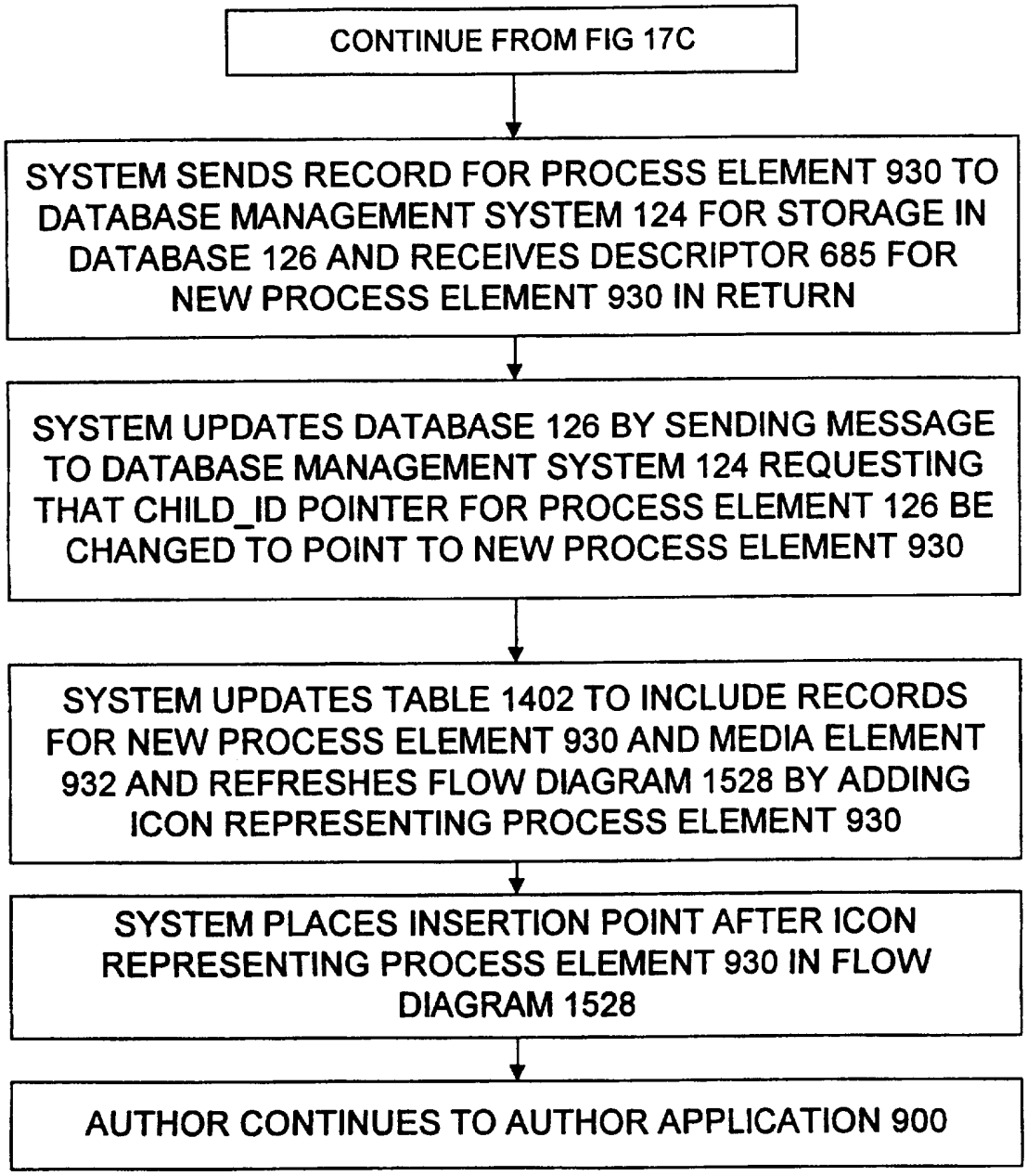

CONTINUE FROM FIG 17C
SYSTEM SENDS RECORD FOR PROCESS ELEMENT 930 TO DATABASE MANAGEMENT SYSTEM 124 FOR STORAGE IN DATABASE 126 AND RECEIVES DESCRIPTOR 685 FOR NEW PROCESS ELEMENT 930 IN RETURN
SYSTEM UPDATES DATABASE 126 BY SENDING MESSAGE TO DATABASE MANAGEMENT SYSTEM 124 REQUESTING THAT CHILD_ID POINTER FOR PROCESS ELEMENT 126 BE CHANGED TO POINT TO NEW PROCESS ELEMENT 930
SYSTEM UPDATES TABLE 1402 TO INCLUDE RECORDS FOR NEW PROCESS ELEMENT 930 AND MEDIA ELEMENT 932 AND REFRESHES FLOW DIAGRAM 1528 BY ADDING ICON REPRESENTING PROCESS ELEMENT 930
SYSTEM PLACES INSERTION POINT AFTER ICON REPRESENTING PROCESS ELEMENT 930 IN FLOW DIAGRAM 1528
AUTHOR CONTINUES TO AUTHOR APPLICATION 900

FIG. 17D

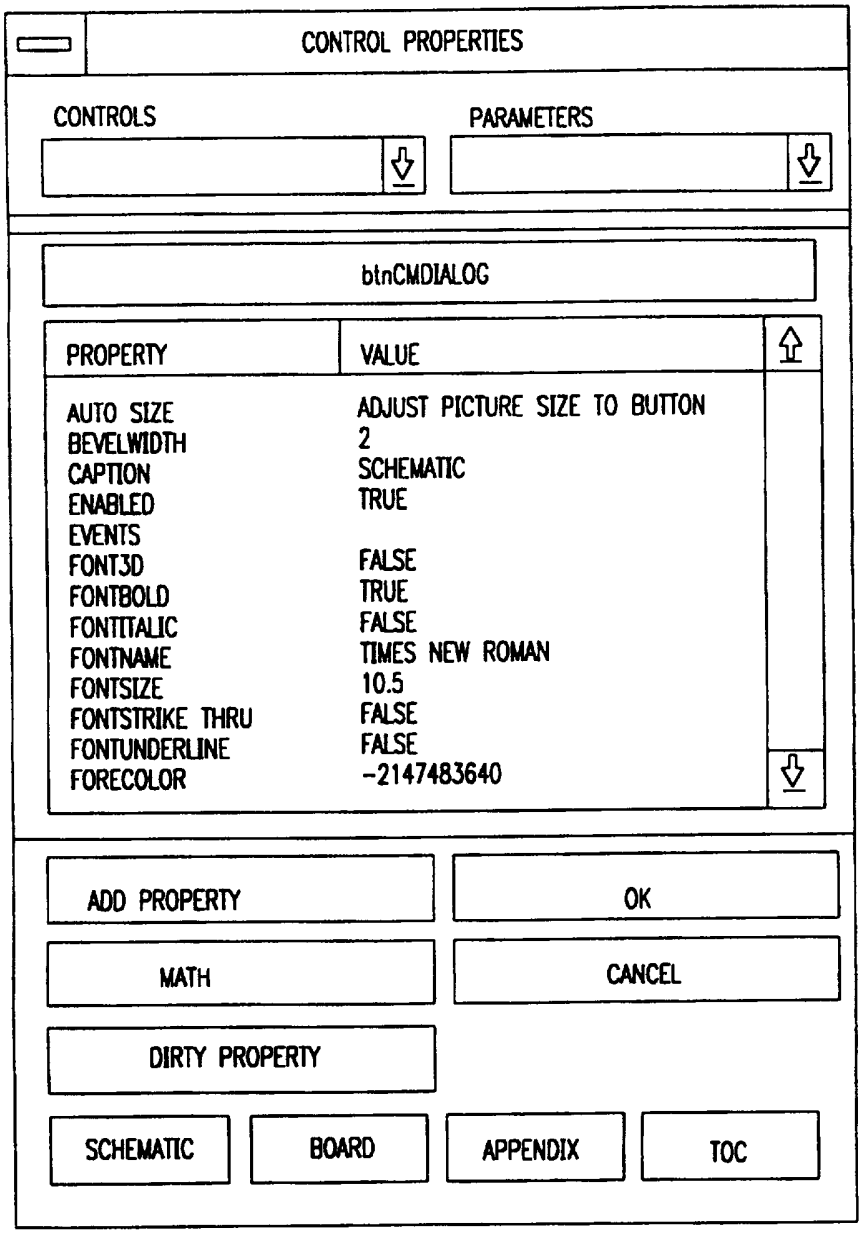

CONTROL PROPERTIES
CONTROLS
PARAMETERS
btnCMDIALOG
PROPERTY
VALUE
AUTO SIZE
ADJUST PICTURE SIZE TO BUTTON
BEVELWIDTH
2
CAPTION
SCHEMATIC
ENABLED
TRUE
EVENTS
FONT3D
FALSE
FONTBOLD
TRUE
FONTITALIC
FALSE
FONTNAME
TIMES NEW ROMAN
FONTSIZE
10.5
FONTSTRIKE THRU
FALSE
FONTUNDERLINE
FALSE
FORECOLOR
-2147483640
ADD PROPERTY
OK
MATH
CANCEL
DIRTY PROPERTY
SCHEMATIC
BOARD
APPENDIX
TOC

FIG. 18

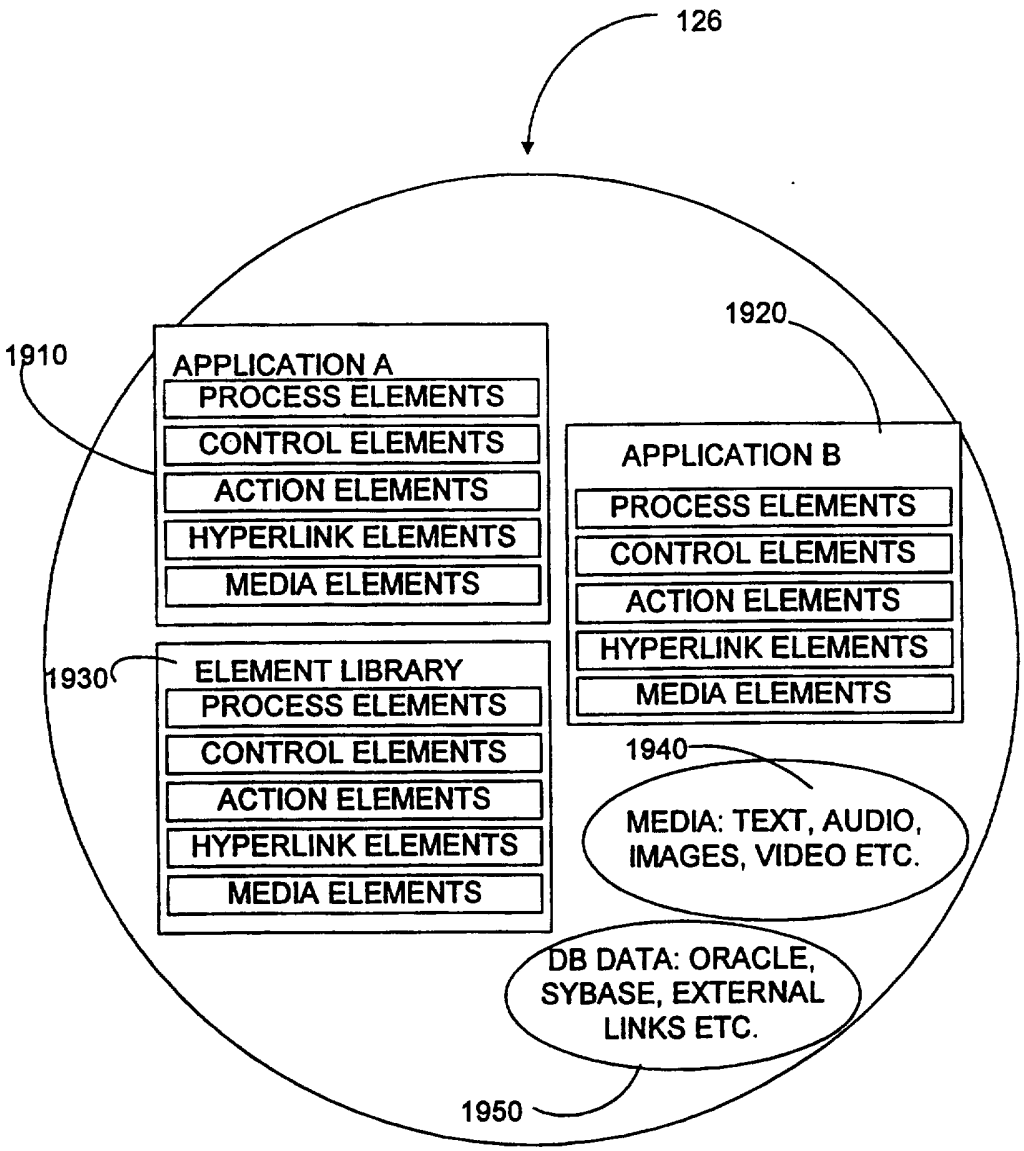

126
1910
APPLICATION A
PROCESS ELEMENTS
CONTROL ELEMENTS
ACTION ELEMENTS
HYPERLINK ELEMENTS
MEDIA ELEMENTS
1920
APPLICATION B
PROCESS ELEMENTS
CONTROL ELEMENTS
ACTION ELEMENTS
HYPERLINK ELEMENTS
MEDIA ELEMENTS
1930
ELEMENT LIBRARY
PROCESS ELEMENTS
CONTROL ELEMENTS
ACTION ELEMENTS
HYPERLINK ELEMENTS
MEDIA ELEMENTS
1940
MEDIA: TEXT, AUDIO, IMAGES, VIDEO ETC.
DB DATA: ORACLE, SYBASE, EXTERNAL LINKS ETC.
1950

FIG. 19

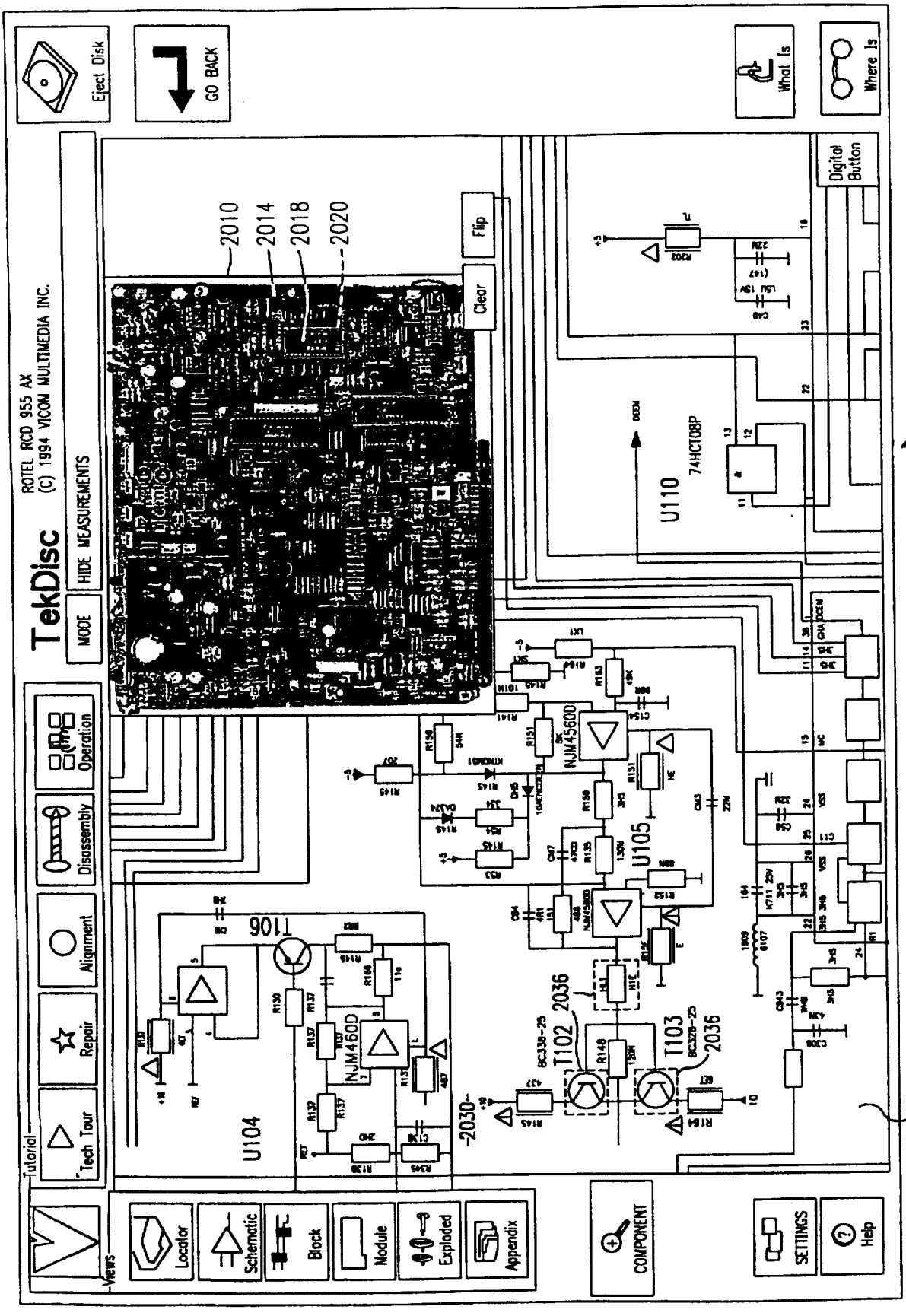

Eject Disk
GO BACK
What Is
Where Is
TekDisc
ROTEL RCD 955 AX
(C) 1994 VICOM MULTIMEDIA INC.
MODE
HIDE MEASUREMENTS
Digital Button
Flip
Clear
2010
2014
2018
2020
2000
U110
74HCT08P
U105
NJM4560D
U104
T102
T103
2036
2030
2032
Tutorial
Tech Tour
Repair
Alignment
Disassembly
Operation
Views
Locator
Schematic
Block
Module
Exploded
Appendix
COMPONENT
SETTINGS
Help

FIG. 20A

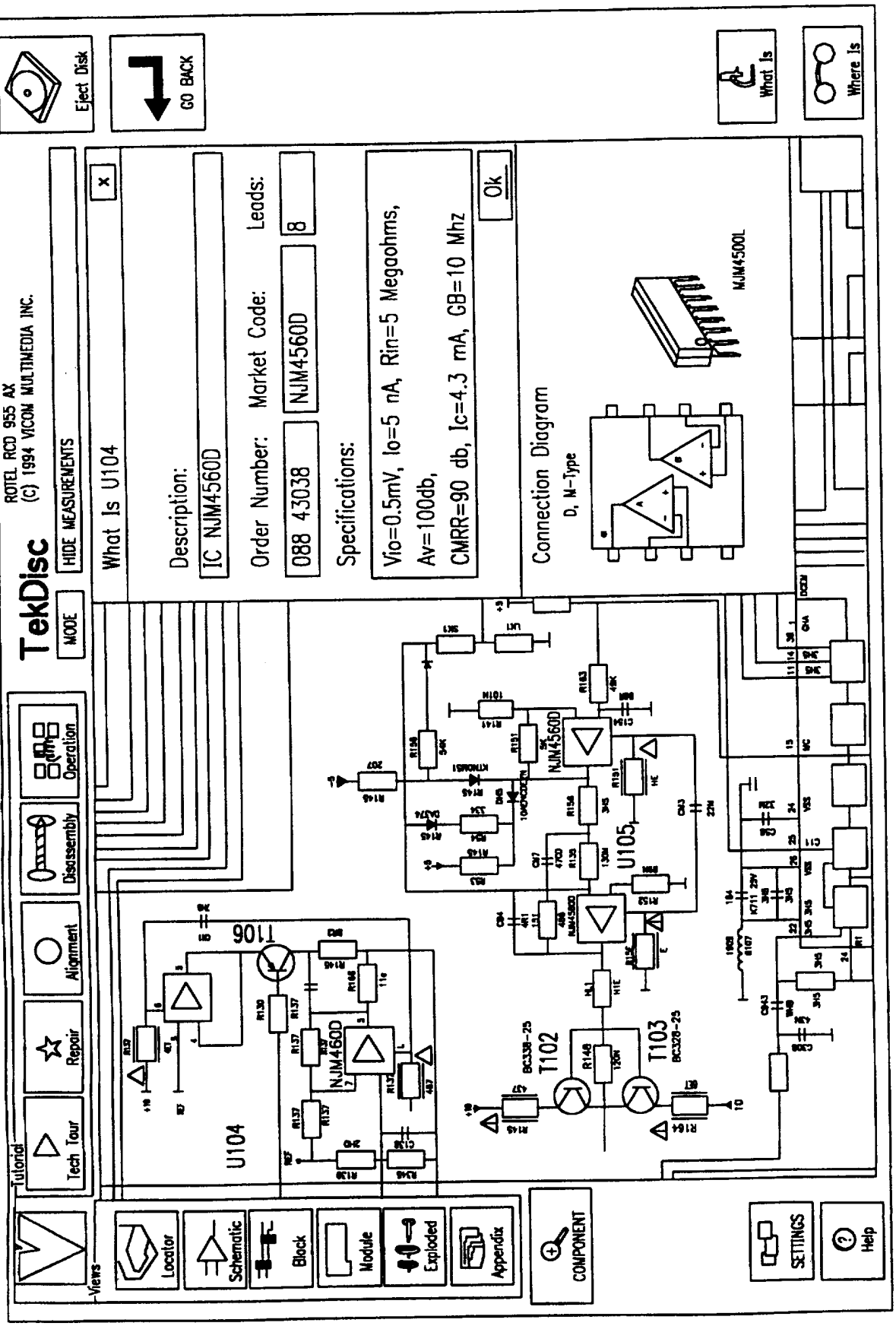

Eject Disk
GO BACK
What Is
Where Is
TekDisc
ROTEL RCD 955 AX
(C) 1994 VICOM MULTIMEDIA INC.
MODE
HIDE MEASUREMENTS
What Is U104
Description:
IC NJM4560D
Order Number:
088 43038
Market Code:
NJM4560D
Leads:
8
Specifications:
Vio=0.5mV, Io=5 nA, Rin=5 Megaohms,
Av=100db,
CMRR=90 db, Ic=4.3 mA, GB=10 Mhz
Ok
Connection Diagram
D, N-Type
NJM4500L
Tutorial
Tech Tour
Repair
Alignment
Disassembly
Operation
Views
Locator
Schematic
Block
Module
Exploded
Appendix
COMPONENT
SETTINGS
Help
U104
U105
T102
T103
NJM4560D

FIG. 20B

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/CA 97/00039

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

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)
IPC 6 G06F

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 practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 94 28480 A (IMAGINE MULTIMEDIA INC) 8 December 1994 see the whole document ---	1,20,30, 35,36, 43,51-53
A	EP 0 578 391 A (WNM VENTURES INC) 12 January 1994 see the whole document ---	1,20,30, 35,36, 43,51-53
A	EP 0 669 587 A (AT & T CORP) 30 August 1995 see the whole document --- -/--	1,20,30, 35,36, 43,51-53

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"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)

"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

"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

"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

"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.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 June 1997	12. 06. 97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Katerbau, R

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

INTERNATIONAL SEARCH REPORT

International Application No
PCT/CA 97/00039

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 94 08310 A (QUARK INC) 14 April 1994 see page 1, line 1 - page 10, line 17	1,20,30, 35,36, 43,51,53

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/CA 97/00039

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9428480 A	08-12-94	AU 7093394 A	20-12-94
EP 0578391 A	12-01-94	US 5440677 A	08-08-95
		JP 7006565 A	10-01-95
EP 0669587 A	30-08-95	CA 2140850 A	25-08-95
WO 9408310 A	14-04-94	AU 5294293 A	26-04-94
		CA 2145765 A	14-04-94
		EP 0663090 A	19-07-95