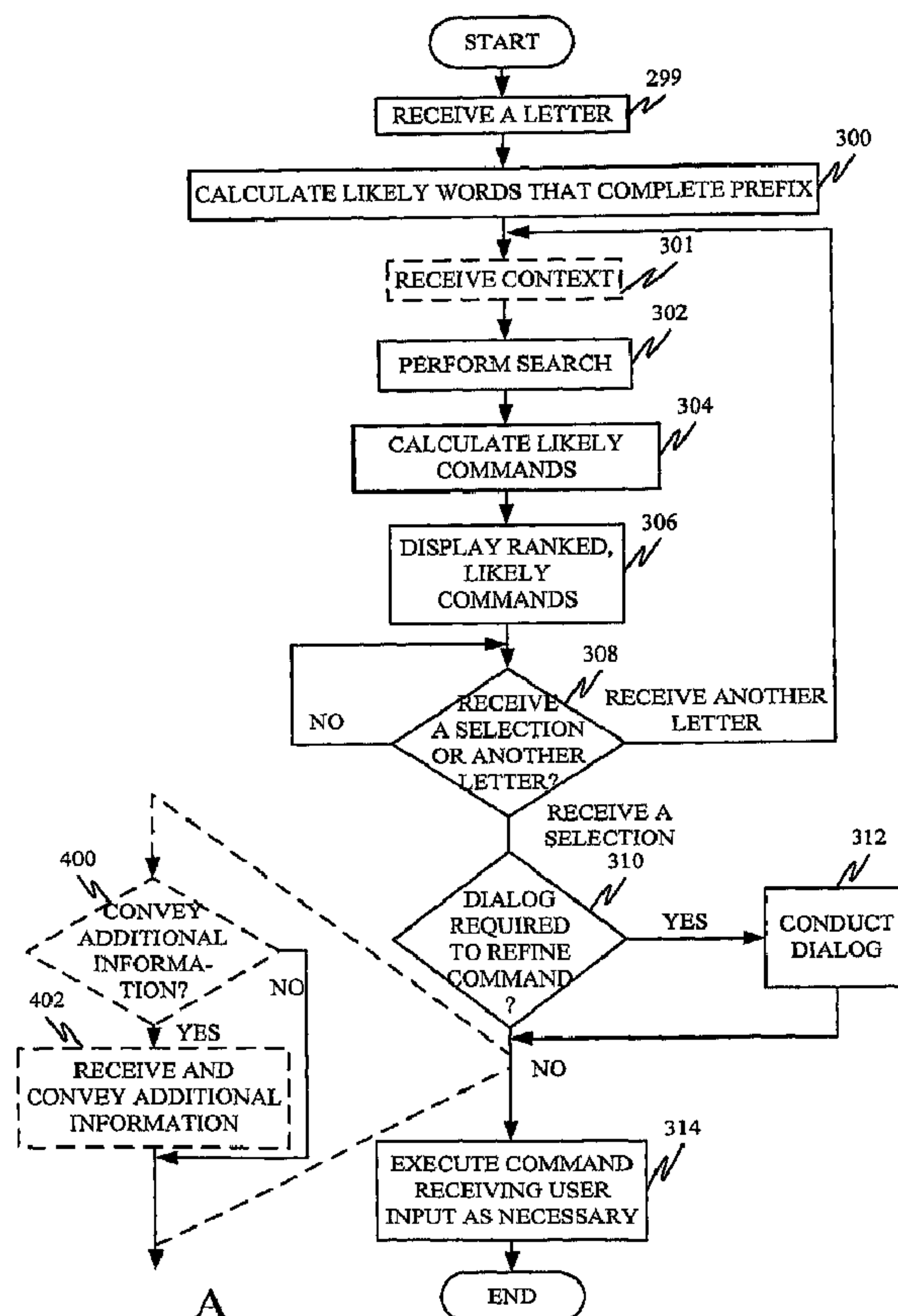




(86) Date de dépôt PCT/PCT Filing Date: 2007/02/23
(87) Date publication PCT/PCT Publication Date: 2007/09/20
(45) Date de délivrance/Issue Date: 2015/04/07
(85) Entrée phase nationale/National Entry: 2008/08/26
(86) N° demande PCT/PCT Application No.: US 2007/004699
(87) N° publication PCT/PCT Publication No.: 2007/106315
(30) Priorité/Priority: 2006/03/10 (US11/372,545)

(51) Cl.Int./Int.Cl. *G06F 9/00* (2006.01),
G06F 17/00 (2006.01)
(72) Inventeurs/Inventors:
BALA, ARAVIND, US;
FRIES, KAREN E., US
(73) Propriétaire/Owner:
MICROSOFT CORPORATION, US
(74) Agent: SMART & BIGGAR

(54) Titre : RECHERCHE D'INSTRUCTIONS S'EXECUTANT DANS DES APPLICATIONS
(54) Title: SEARCHING FOR COMMANDS TO EXECUTE IN APPLICATIONS



(57) Abrégé/Abstract:

A user can access a searching component that allow the user to enter search terms to search for command associated with a computer program. The most likely command are provided to the user, and a command can be selected for execution.

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

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
20 September 2007 (20.09.2007)

PCT

(10) International Publication Number
WO 2007/106315 A3

(51) International Patent Classification:

G06F 9/00 (2006.01) G06F 17/00 (2006.01)

(21) International Application Number:

PCT/US2007/004699

(22) International Filing Date:

23 February 2007 (23.02.2007)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

11/372,545 10 March 2006 (10.03.2006) US

(71) Applicant (for all designated States except US): MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventor: BALA, Aravind; One Microsoft Way, Redmond, WA 98052-6399 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN,

CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

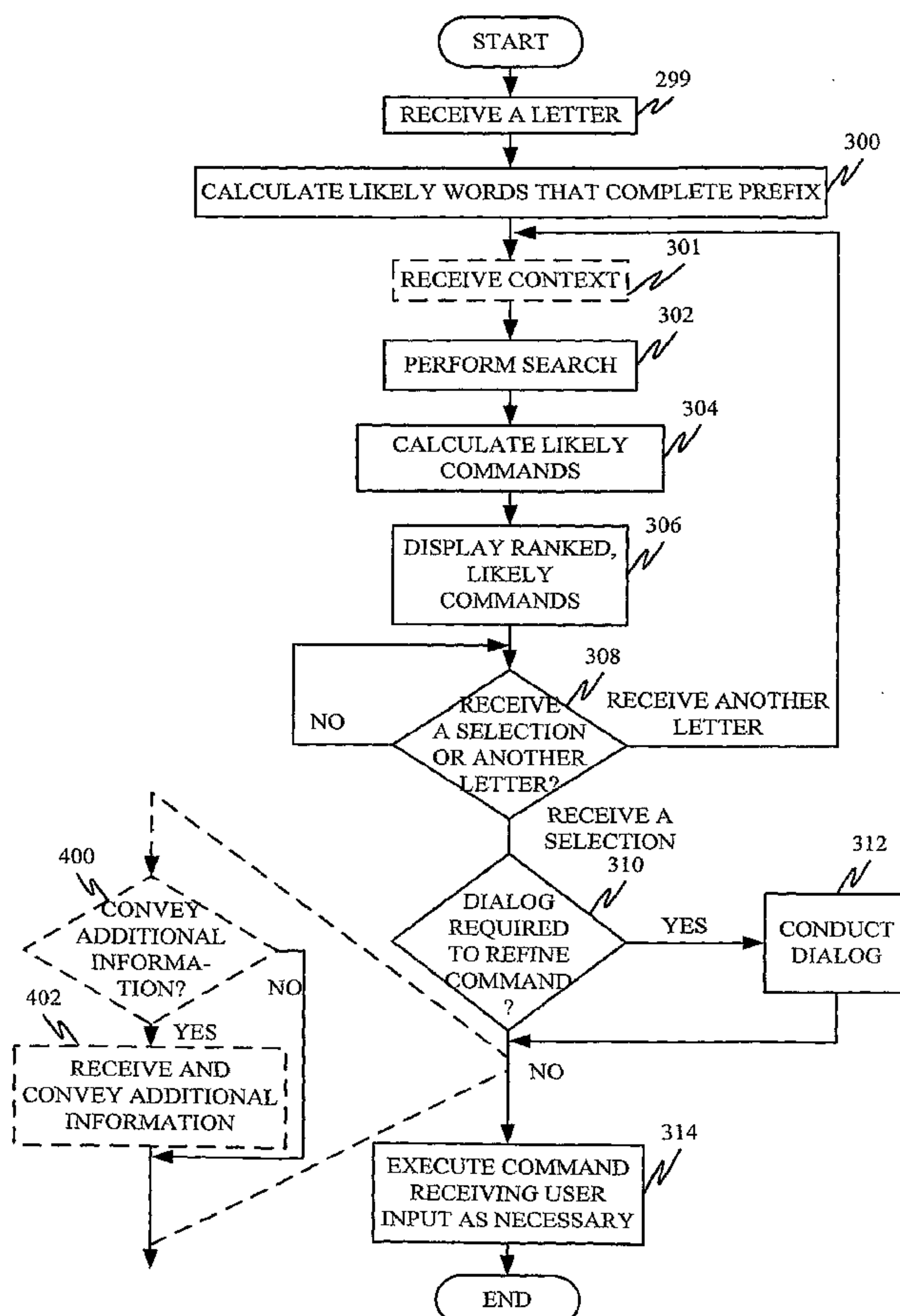
(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))

[Continued on next page]

(54) Title: SEARCHING FOR COMMANDS TO EXECUTE IN APPLICATIONS



(57) Abstract: A user can access a searching component that allow the user to enter search terms to search for command associated with a computer program. The most likely command are provided to the user, and a command can be selected for execution.

WO 2007/106315 A3

WO 2007/106315 A3



Published:

- *with international search report*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(88) Date of publication of the international search report:

8 November 2007

SEARCHING FOR COMMANDS TO EXECUTE IN APPLICATIONS

BACKGROUND

5 A wide variety of different types of computer applications are currently in use. Such computer applications often fall into three or more different categories. For instance, rich client applications are typically applications that run on a client device, such
10 as a desktop computer or laptop computer. Other software applications include web-based applications in which a client interacts with a web-based device, such as a server using a web browser, in order to run the application. Still other software applications include mobile device
15 applications. Mobile device applications might run on mobile devices, such as personal digital assistants (PDAs), telephones, handheld computers, etc.

 In running a software application, a user typically wishes to perform one or more different tasks.
20 As used herein, the term task means anything that a user wishes to do in the context of a given software application.

 In order to allow users to accomplish the desired tasks, many current software applications provide
25 a graphical user interface. The user executes one or more commands on the graphical user interface to accomplish a desired task. There are substantially three main ways by which people accomplish tasks using a graphical user interface, and they depend on how frequently the user
30 performs a given task. For tasks that are performed frequently by the user, the user might learn or memorize the sequence of steps (or commands) required to perform the task with the graphical user interface. Also with frequently performed tasks, applications might expose the

user interface elements for performing the most popular tasks prominently in the user interface.

A second group of tasks are those that are performed infrequently by the user. The user may typically browse the menus or tool bars displayed by the graphical user interface, in order to attempt to locate the user interface element that can be used to perform the desired task. Another way in which users perform infrequent tasks is to seek the help of other people to find out how to perform the task. In doing this, the user might go to a news group or simply ask another individual how to perform the task.

Finally, when the user does not remember how to perform the task, and cannot find out how to perform the task by asking others, users might typically invoke a help mechanism associated with the application. For instance, some applications are associated with help documentation stored on a client, or on a web site. The help documentation allows a user to search for a help topic that often explains what to do in order to accomplish a task that a user wishes to perform.

Of course, there are a number of problems associated with these types of current systems. Unless the user has the steps for performing a task memorized, it can be fairly cumbersome for the user to find out how to perform the task. Asking news groups or physically asking other persons how to perform a task takes a relatively large amount of time and thus reduces efficiency.

In addition, even where the user attempts to hunt through the user interface to find the user interface element necessary to accomplish the desired task, the user may encounter problems. For instance, in most graphical user interfaces, the features or elements associated with the graphical user interface are categorized under other

elements. In other words, the user may go to the "File" drop down menu to look for a particular feature or element of the graphical user interface. The user may also go to the "Edit" drop down menu to look for another feature. Of course, if the sought after feature is under an unexpected drop down menu, this can result in frustration and extra time required for the user to find that feature.

Similarly, attempting to determine how to perform a task using help documentation often requires a fairly high level of knowledge about the task. For instance, some help documentation is simply an alphabetical listing of different tasks. This may be extremely unhelpful to a user who does not know the technical term for the task to be performed. By way of example, in one spreadsheet software application, adding rows and columns to a spreadsheet is performed using an "insert" command. Of course, it would take a user a fairly long time to track this command down, if the user was simply looking under the "add" commands in the help documentation.

Similarly, many help mechanisms provide information that is not useful in a given context. For instance, assume the user is in a word processing application which has no tables in it, but the user wishes to add a table. The help mechanism might typically be arranged with all information related to tables found in a plurality of different spots within the help documentation. Thus, the help documentation might include modifying existing tables, adding or deleting rows or columns from tables, and a wide variety of other information dealing with already-existing tables. Of course, since the user's document has no already-existing tables, this information is completely useless to the user. Yet, the user must sift through this information in

51039-28

order to identify the steps necessary to add a table.

The discussion above is merely provided for general background information and is not intended to be used as an aid in determining the scope of the claimed subject matter.

5

SUMMARY

A user can access a searching component that allows the user to enter search terms to search for commands associated with a computer program. In one embodiment, the search component is context sensitive so that the search results are dependent not only upon the search terms input by the user, but upon the current
10 context of the application. In one embodiment, the search is performed by computing likely search results after each letter entered by the user in a search term. In one embodiment, a search is not only conducted for the desired commands, but the command is automatically executed as well. In another embodiment, a search returns relevant commands and a user can select a desired command from a list.

15

According to one aspect of the present invention, there is provided a computer implemented method of identifying commands in an application, comprising: receiving a search request indicative of a desired task to be performed with the application; searching, with a processor, a command store, that stores commands for the application, using the search request; identifying, with a processor,
20 as possible commands to perform the desired task, a subset of the commands in the command store based on the search request, the subset including a plurality of different commands for performing a plurality of different tasks in the application; returning, with a processor, representations of the possible commands in a user actuable form such that one of the possible commands that corresponds to the
25 desired task can be actuated by a user to execute the desired task in the application, wherein returning representations of the possible commands comprises returning representations of the possible commands in ranked order based on a score calculated for each possible command in the subset, the score being indicative of

51039-28

how likely it is that the possible command can be used to perform the desired task: receiving a user selection of a given one of the representations of possible commands, the given one of the representations of possible commands representing at least two specific possible commands, each specific possible command

5 representing a command bundle of a plurality of subcommands; conducting additional dialog, prior to performing the plurality of subcommands in the command bundle, to obtain information necessary to disambiguate between, and identify one of, the at least two specific possible commands, and to perform all of the subcommands in the command bundle, in response to receiving the user selection of the given one of the

10 representations of the possible commands, the additional dialog being conducted utilizing a guided help screen that includes an indication of the given one of the representations of possible commands and tutorial content associated with the given one of the representations of possible commands, the tutorial content including at least three sections, a first section of the tutorial content providing textual information

15 about the given one of the representations of possible commands, a second section of the tutorial content providing a list of user selectable options corresponding to the at least two specific possible commands, and a third section of the tutorial content providing a combination of textual and graphical information about the given one of the representations of possible commands, the user selection of the given one of the

20 representations of the possible commands including receiving a user selection of one of the options from the list of user selectable options in the second section of the tutorial content; and automatically performing all of the subcommands in the command bundle without requiring additional information from the user.

According to another aspect of the present invention, there is provided

25 a computer-readable medium having computer-executable instructions stored thereon that, when executed by a computer, cause the computer to implement the method as described above or below.

According to still another aspect of the present invention, there is provided a method of identifying commands in an application, comprising: receiving

30 a search request indicative of a desired task to be performed with the application;

55450-8

searching a command store, that stores commands for the application, using the search request; identifying as possible commands to perform the desired task, a subset of the commands in the command store based on the search request, the subset including a plurality of different commands for performing a plurality of different tasks in the application; and returning the possible commands in a user actuable form such that one of the possible commands that corresponds to the desired task can be actuated by a user to execute the desired task in the application, wherein returning the possible commands comprises returning the possible commands in ranked order based on a score calculated for each possible command in the subset, the score being indicative of how likely it is that the possible command can be used to perform the desired task.

According to yet another aspect of the present invention, there is provided a system for locating a command to perform a task in any of a plurality of different applications, comprising: a processor; and a memory having processor-executable instructions stored thereon that, when executed by the processor, implement a searching component configured to receive a search request and a current context of an application in which the command is to be executed and to identify, from a plurality of commands, a plurality of possible commands, comprising a subset of the plurality of commands, based on the search request and the context, the plurality of possible commands each actuable to perform a different task, and to return and display the plurality of possible commands in user selectable form, such that a selected one of the plurality of possible commands can be executed from the user selectable form to perform the task in a given one of the plurality of different applications.

According to a further aspect of the present invention, there is provided a computer-readable medium having computer-executable instructions stored thereon which, when executed by a computer, cause the computer to perform steps of: receiving a command search request; receiving context information indicative of a current context of an application for which a command is sought; searching a data store to identify a plurality of possible commands based on the command search

55450-8

request and the context information, the possible commands identified changing with changes in the context information; and returning a list of the plurality of possible commands, each actuable to perform a different task, as user actuable graphical user interface elements, ordered according to a relationship between the possible
5 commands and the search request and the context information.

This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the
10 claimed subject matter. The claimed subject matter is not limited to implementations that solve any or all disadvantages noted in the background.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one illustrative embodiment of a computing environment with which the present subject matter can be used.

FIG. 2 is a block diagram of one illustrative mobile device.

FIG. 3 is a block diagram in which searching and command processing are performed on a client.

5 FIG. 4 is a flow diagram illustrating one illustrative embodiment of the operation of the system shown in FIG. 2.

FIG. 4 is a flow diagram illustrating a number of optional steps.

10 FIGS. 5-10B are screenshots showing embodiments of graphical user interfaces illustrating features of the present subject matter.

FIG. 11 is a block diagram showing command searching and processing being performed on a server.

15 FIG. 12 is a block diagram showing command searching and processing being performed on a server which supports a server-based application.

DETAILED DESCRIPTION

The present subject matter deals with searching
20 for commands associated with software applications, and executing those commands. However, before that subject matter is discussed in more detail, two illustrative computing devices will be described.

FIG. 1 illustrates an example of a suitable
25 computing system environment 100 on which embodiments may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the claimed subject matter.
30 Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

Embodiments are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations
5 that may be suitable for use with various embodiments include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs,
10 minicomputers, mainframe computers, telephony systems, distributed computing environments that include any of the above systems or devices, and the like.

Embodiments may be described in the general context of computer-executable instructions, such as
15 program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Some embodiments are designed to be practiced in distributed
20 computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules are located in both local and remote computer storage media including memory storage
25 devices.

With reference to FIG. 1, an exemplary system for implementing some embodiments includes a general-purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not
30 limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory

controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and

not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should
5 also be included within the scope of computer readable media.

The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access
10 memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are
15 immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

20 The computer 110 may also include other removable/non-removable volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk
25 drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile
30 computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is

typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable
5 memory interface, such as interface 150.

The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the
10 computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134,
15 application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies.

20 A user may enter commands and information into the computer 110 through input devices such as a keyboard 162, a microphone 163, and a pointing device 161, such as a mouse, trackball or touch pad. Other input devices (not shown) may include a joystick, game pad, satellite dish,
25 scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial
30 bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which

may be connected through an output peripheral interface 195.

The computer 110 is operated in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a handheld device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on remote computer 180. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

FIG. 2 is a block diagram of a mobile device 200, which is an exemplary computing environment. Mobile

device 200 includes a microprocessor 202, memory 204, input/output (I/O) components 206, and a communication interface 208 for communicating with remote computers or other mobile devices. In one embodiment, the afore-
5 mentioned components are coupled for communication with one another over a suitable bus 210.

Memory 204 is implemented as non-volatile electronic memory such as random access memory (RAM) with a battery back-up module (not shown) such that information
10 stored in memory 204 is not lost when the general power to mobile device 200 is shut down. A portion of memory 204 is illustratively allocated as addressable memory for program execution, while another portion of memory 204 is illustratively used for storage, such as to simulate
15 storage on a disk drive.

Memory 204 includes an operating system 212, application programs 214 as well as an object store 216. During operation, operating system 212 is illustratively executed by processor 202 from memory 204. Operating
20 system 212 is illustratively designed for mobile devices, and implements database features that can be utilized by applications 214 through a set of exposed application programming interfaces and methods. The objects in object store 216 are maintained by applications 214 and operating
25 system 212, at least partially in response to calls to the exposed application programming interfaces and methods.

Communication interface 208 represents numerous devices and technologies that allow mobile device 200 to send and receive information. The devices include wired
30 and wireless modems, satellite receivers and broadcast tuners to name a few. Mobile device 200 can also be directly connected to a computer to exchange data therewith. In such cases, communication interface 208 can be an infrared transceiver or a serial or parallel

communication connection, all of which are capable of transmitting streaming information.

Input/output components 206 include a variety of input devices such as a touch-sensitive screen, buttons, rollers, and a microphone as well as a variety of output devices including an audio generator, a vibrating device, and a display. The devices listed above are by way of example and need not all be present on mobile device 200. In addition, other input/output devices may be attached to or found with mobile device 200.

FIG. 3 is a block diagram of a computing environment in accordance with an embodiment in which command searching and processing is performed on a client 250. Client 250 is illustratively a desktop computer, a laptop computer, or a mobile device, such as a personal digital assistant, a telephone, a handheld computer, etc. In the embodiment shown in FIG. 3, client 250 has a software application 252 thereon which is accessed by user 254 such that the user can execute one or more commands to perform a variety of different tasks. For example, application 252 may be a word processing program, a spreadsheet program, a presentation program, or any other software application that can be accessed by a user 254, through a user interface 271, to perform tasks.

Client 250 is shown with a platform 256. In one embodiment, platform 256 is, for example, an operating system that supports a plurality of different applications 252. In the embodiment shown in FIG. 3, platform 256 includes command searching and processing component 258 and command store 259. As is described in greater detail below, component 258 is exposed to user 254 through application 252 (or independent of application 252) such that the user can search for commands associated with application 252, and optionally execute the desired

commands. In one embodiment, platform 256 exposes an interface 260, such as an application programming interface, which can be invoked by application 252.

Client 250 is also shown with an optional speech
5 recognition component 270. This can be used as described below.

In the embodiment shown in FIG. 3, client 250 is also connected to a server 262 through a network 264. Network 264 can be any desired network, such as a wide
10 area network, a local area network, etc.

FIG. 4 is a flow diagram illustrating one embodiment of the operation of the system shown in FIG. 3. FIGS. 3 and 4 will now be described in conjunction with one another. For the purposes of this discussion, it is
15 assumed that user 254 launches application 252 and desires to perform a task using application 252. The user illustratively does not know the command or commands required to perform the task. In other words, the user has not memorized the steps necessary (or the location of
20 the user interface elements required) to perform the desired task. Therefore, user 254 must locate the command required to perform the desired task.

In accordance with one embodiment, user 254 begins to enter a search request. In one embodiment, user
25 254 can enter the search request through speech recognition component 270, by simply speaking the desired search request. For purposes of the present discussion, however, it will be assumed that the user types in the search request one letter at a time, but the invention is
30 not to be so limited.

In any case, FIG. 5 shows one graphical user interface 271 which can be used to receive entry of the search request. User interface 271 in FIG. 4 illustrates that application 252 is a presentation program referred to

as "Program A". Command searching and processing component 258 exposes text box 274 through user interface 271 of application 252, to the user. The user can enter a search request, letter by letter, into text box 274.

5 Therefore, after the user 254 types in a first letter in text box 274 on user interface 271, the letter is transmitted, as a search request 280, to command searching and processing component 258 in platform 256. In order to do this, in one embodiment, application 250
10 provides the search request 280 through the interface 260 exposed by component 258. Receiving the first letter at command searching and processing component 258 is indicated by block 299 in FIG. 4.

 Component 258 then calculates likely words that
15 complete the letter string entered thus far. For instance, if the user has typed "ins" the most likely completion of this prefix might be "insert". This is indicated by block 300 in FIG. 4 and is described in more detail below with respect to the example shown in FIG. 5.

20 In one embodiment, component 258 also receives the context 281 of application 252. For instance, the context will illustratively identify the particular application 252 for which a command is sought, and provide an indication of the particular contents of the window
25 currently open and under focus in the application, a position of the cursor relative to the elements on the open window, etc. Examples of context 281 are described in greater detail below and the optional step of receiving the context is indicated by block 301 in FIG. 4.

30 Once the first letter is received, likely completions are calculated, and the context component 258 performs a search through command store 259 that stores the commands associated with a plurality of different applications. This search is performed based on the

received letters, likely completions and the context. Performing the search is indicated by block 302 in FIG. 4.

5 In performing the search, component 258 identifies possible commands and calculates a score indicative of how likely it is that each of the possible commands is being requested by the user in the search request. This is indicated by block 304 in FIG. 4. While the search can be performed in any desired way, in one embodiment, the search is performed using a search engine
10 that is trained with prior probabilities of commands.

Once the calculation has been made, component 258 returns the search results through interface 260 where they are displayed at user interface 271 through application 252. Of course, the results can be returned
15 without going through application 252 as well. In any case, in one embodiment, component 258 not only displays the most likely commands given the search request, but also displays them in ranked order, given the score calculated for each command in block 204. Displaying the
20 ranked commands is indicated by block 306 in FIG. 4.

User 254 can then either select one of the displayed commands, or continue typing additional letters in the search request. This is indicated by block 308 in FIG. 4. In the embodiment in which the user does not
25 select one of the commands, but instead continues to type letters, processing reverts back to block 302 and another search is performed, scores are calculated, and the likely commands are again displayed in ranked order, as indicated by blocks 304 and 306.

30 Assume for the sake of the example shown in FIG. 5 that the user 254 wishes to insert a table in the particular application 252 which is open. In the embodiment shown in FIG. 5, user interface 271 shows that user 254 has typed into text box 274 the letters "ins".

The context has also been provided to component 258. The context indicates, among other things, that the open application 252 is a presentation application. Component 258 calculates the probabilities of commands associated with the input "ins" stored in command store 259. In the particular example shown in FIG. 5, component 258 determines that the most probable completion of "ins" is "insert", and component 258 displays, as search results, commands which relate to inserting or adding in the given context.

It will be noted from this example that component 258 illustratively not only searches based upon the particular letters input, but based upon synonyms of the most likely completions of those letters. In other words, the most likely completion of the letters "ins" in the present context is "insert". Therefore, component 258 identifies synonyms for "insert" (such as "add") and searches for commands that are for "inserting" or "adding" things and that would be most likely, given the current context of the application 252.

As shown in FIG. 5, component 258 returns the search results, ranking them in order. It can be seen that the first command returned is "insert slide" which is more probable than any of the other commands in the list. The other commands in the list are rank ordered, according to the score calculated by component 258 (which can be a likelihood or probability or other desired measure) and they are numbered, according to their rank. In the embodiment shown in FIG. 5, the first nine results are numbered according to their rank, and a plurality of other commands are also provided, but are not numbered.

The user can select one of the commands from the list of returned results 275 by simply clicking on the displayed command. However, in one embodiment, the user

can also select one of the numbered search results by simply typing that number in box 274. Any other desired way of selecting a command from the list 275 can be used as well.

5 In accordance with the example shown in FIG. 5, the user selects the "insert table" command. Having component 258 receive a command selection is indicated by block 308 in FIG. 4..

10 Once the user has selected a command from list 275, if more dialog with the user is required in order to further refine or disambiguate the command, component 258 can conduct that additional dialog. This is discussed in greater detail below with respect to FIG. 6. Determining whether the extra dialog is required and conducting that
15 dialog is indicated by blocks 310 and 312 in FIG. 4.

 Once the dialog is complete, or if no further dialog is required, that means that the selected command has been fully disambiguated by component 258. Component 258 can then either provide additional information to user
20 254 about the selected command (such as tutorial information), or component 258 can simply execute the command selected by the user. Providing the additional information is described in more detail below with respect to FIGS. 4A, 8 and 9, and executing the selected command
25 is indicated by block 314 in FIG. 4.

 As an example of executing the commands, where the user has selected the "insert table" command, component 258 (after it determines that no additional dialog is necessary) controls application 252 such that
30 application 252 takes the user to the appropriate place in the application to execute the "insert table" command. FIG. 5 displays the appropriate user interface mechanism that allows the user to execute the "insert table" command.

As an example of conducting additional dialog to refine a command, component 258 may conduct a dialog with the user as at block 312 in FIG. 4 to determine the size of the table which the user wishes to add. Component 258
5 then adds it automatically. In another embodiment, component 258 simply allows the user to walk through the "insert table" functionality provided by application 252, once component 258 has taken the user to that place in the user interface provided by application 252.

10 Also, the additional dialog may further define what the user wishes to do. For example, if the user inputs "delete cell and table", the component 258 may respond with an inquiry such as "which table" or "which cell", and then delete the specified table or cell based
15 on the response from the user.

In accordance with another embodiment, component 258 (or another component invoked by component 258) automatically performs the command selected by the user, receiving user inputs where necessary. For instance, user
20 254 can select the desired user interface elements to insert a table from the list 275 of possible commands. Component 258 (or another component invoked by component 258) can then assume control of application 252 and begin to automatically perform that command. However, when it
25 comes to the point where the user interface asks the user to specify the size of the table (in terms of columns and rows) component 258 simply highlights the field to be modified by the user and pauses until the user has input the desired information and clicked "OK". In that
30 example, component 258 (or other component) simply waits until the user has selected the desired number of columns and rows and clicked "OK" and component 258 (or other component) then walks the application 252 through the remainder of the steps necessary to perform the command of

inserting the desired size table within the application, at the specified spot. Automatically executing the command is indicated by block 314 in FIG. 4.

Of course, if component 258 is not configured to automatically execute the command selected by the user, the user can continue to use component 258 as the user is taking the steps to execute the command. For instance, if, in FIG. 6, the user again types "ins" into text box 274, component 258 again identifies the possible commands and calculates the scores for the various commands and again displays a list 275 of possible commands, given the new context. This is shown in FIG. 7. Specifically, FIG. 7 illustrates that component 258 has now calculated the highest scores for the commands of inserting rows and inserting columns in a table, because the context indicates that the table has already been added as shown in FIGS. 5 and 6.

As briefly mentioned above, component 258 can convey additional information to the user, and not just execute the command. This is indicated in FIG. 4A. For instance, component 258 may illustratively be configured to provide user 254 with tutorial information or guided help in executing the command. In doing so, component 258 first determines that it must convey this additional information (e.g., the guided help) to the user as indicated at block 400. The decision to provide the additional information can be predetermined by the developer, or set by the user using any desired user interface element, or based on any other desired criteria. Component 258 then retrieves and conveys the additional information, as indicated by block 402. In the embodiment discussed herein, the additional information is the guided help or tutorial help, which can be offered to a user in

executing a command, and an example of this is illustrated in FIGS. 8 and 9.

For instance, assume for the sake of example that the user types "back" in text box 274 as shown in FIG. 8. Component 258 then calculates the most probable commands, given the input "back" and given the present context and provides the ranked results shown in list 275. FIG. 8 also shows that the user has selected the first result in the list (i.e., the "change the background [wizard]" command). The selected command is actually comprised of a command bundle of subcommands that are executed to perform the overall command or task of changing the background. Component 258 then launches the "change background" command bundle associated with application 252. In the embodiment illustrated, one example of the guided help is shown in FIG. 9.

FIG. 9 is a guided help and command interface 500 which not only provides some tutorial information to the user about executing the selected command, but also provides mechanisms by which the user can actually execute the command. For instance, user interface 500 provides a tutorial display portion 502 that displays tutorial content, as well as a tutorial portion 504 that displays additional tutorial content. User interface 500 also provides command elements 506 that allow a user to select the desired subcommands, within the command bundle, for changing the background of presentation material provided in application 252. The "next" button 506 allows the user to walk through the tutorial and actually execute the command as the user is walking through the tutorial.

From the above discussion, it will be clear that the particular user interface 271 for conducting command searching can take a wide variety of different forms. For instance, FIG. 10A shows another embodiment of user

interface 271. In the embodiment shown in FIG. 10A, the user has entered "open" in text box 274 and the possible commands are shown in list 275. The user can ask for additional results using elements 277.

5 FIG. 10B shows an embodiment of user interface 271 after the user has selected the "open presentation" command. Component 258 then controls application 252 to bring up the "open" file dialog illustrated in FIG. 10B.

 From the description thus far, it is clear that
10 the command searching and processing component 258, and the subject matter related to it, are applicable not only to rich client applications (such as word processing applications, spreadsheets, presentation software, etc.) that reside on a client (such as a desktop or laptop
15 computer) but that the subject matter is also applicable and useful, in an environment in which application 252 resides on a mobile device, such as a personal digital assistant, a mobile telephone, a handheld computer, etc. However, FIGS. 11 and 12 illustrate that the command
20 searching and processing component 258, and the associated subject matter, are also useful in other environments. In one such environment, application 252 resides on a client, but command searching and processing component 258 resides remotely, such as on a server. In another such
25 environment, application 252 and command searching and processing component 258 both reside remotely from the client such as on a server. Items in FIGS. 11 and 12 are numbered similarly to corresponding items in FIG. 3.

 More specifically, FIG. 11 shows that command
30 searching and processing component 258, and command store 259 are no longer resident on client 250. Instead, component 258 resides on server 262, and component 259 is shown coupled to component 258 and can reside at any desirable location. In the embodiment shown in FIG. 11,

search request 280 and context 281 are provided through platform 256, over network 264, to component 258 on server 262. Component 258 performs the search, calculates the scores associated with the results, and provides the
5 ranked search results back through network 264 and platform 256 to application 252, where they are provided to user 254 through user interface 271.

FIG. 12 is similar to FIG. 11, except that application 252 is now also located on server 262.
10 Therefore, application 252 can be a web-based application which is accessed by client 250 over network 264. In the embodiment shown in FIG. 12, the request and search results are transmitted back and forth between application 252 and component 258 at server 262.

15 It can thus be seen that the present system uses language to find substantially any command that a user wishes to execute in an application. The user can provide a search request in the user's own language, and the present system illustratively maps that language to the
20 correct command by searching based on prior probabilities, given context. The commands that are returned are likely the most relevant ones for a given search request, in the current context. Commands returned, once selected, can be automatically performed for the user, or a user interface
25 can be brought up based on the selected command, a wizard can be launched, a rich wizard can be launched, or a tutorial can be run based on the selected commands.

It should also be noted that the present system pivots functionality of the application around the
30 particular command that the user wishes to execute. For instance, if the user wishes to find all commands in the system that have to do with printing, it may be difficult in conventional systems to find those specific commands, because printing information is located in help systems in

many different places relative to many different functions within an application. However, the present component 258 can be used to gather all of that information into a single place, in response to a search request for a
5 printing-related command.

Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not
10 necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

51039-28

CLAIMS:

1. A computer implemented method of identifying commands in an application, comprising:

receiving a search request indicative of a desired task to be performed
5 with the application;

searching, with a processor, a command store, that stores commands for the application, using the search request;

identifying, with a processor, as possible commands to perform the desired task, a subset of the commands in the command store based on the search
10 request, the subset including a plurality of different commands for performing a plurality of different tasks in the application;

returning, with a processor, representations of the possible commands in a user actuable form such that one of the possible commands that corresponds to the desired task can be actuated by a user to execute the desired task in the
15 application, wherein returning representations of the possible commands comprises returning representations of the possible commands in ranked order based on a score calculated for each possible command in the subset, the score being indicative of how likely it is that the possible command can be used to perform the desired task:

receiving a user selection of a given one of the representations of
20 possible commands, the given one of the representations of possible commands representing at least two specific possible commands, each specific possible command representing a command bundle of a plurality of subcommands;

conducting additional dialog, prior to performing the plurality of subcommands in the command bundle, to obtain information necessary to
25 disambiguate between, and identify one of, the at least two specific possible commands, and to perform all of the subcommands in the command bundle, in response to receiving the user selection of the given one of the representations of the

51039-28

possible commands, the additional dialog being conducted utilizing a guided help screen that includes an indication of the given one of the representations of possible commands and tutorial content associated with the given one of the representations of possible commands, the tutorial content including at least three sections, a first
 5 section of the tutorial content providing textual information about the given one of the representations of possible commands, a second section of the tutorial content providing a list of user selectable options corresponding to the at least two specific possible commands, and a third section of the tutorial content providing a combination of textual and graphical information about the given one of the
 10 representations of possible commands, the user selection of the given one of the representations of the possible commands including receiving a user selection of one of the options from the list of user selectable options in the second section of the tutorial content; and

automatically performing all of the subcommands in the command
 15 bundle without requiring additional information from the user.

2. The method of claim 1 wherein identifying possible commands comprises:

receiving a context input indicative of a context of the application and identifying the possible commands based on the context input.

20 3. The method of claim 1 wherein returning the possible commands in a user selectable form comprises:

displaying a user actuatable interface element, for each possible command, that can be actuated by the user through a graphical user interface to perform the possible command.

25 4. The method of claim 1 wherein receiving a search request comprises:
 displaying a text entry mechanism; and

51039-28

receiving a text entry indicative of the search request.

5. The method of claim 4 wherein returning the possible commands in a user selectable form comprises:

5 displaying a user actuable interface element, for each possible command, that can be actuated by the user through a graphical user interface to perform the possible command; and

displaying an associated command identifier, associated with each user actuable interface element, wherein the command identifier can be entered in the text entry mechanism to perform the possible command.

10 6. The method of claim 4 wherein identifying possible commands and returning the possible commands are performed after each character received in the text entry mechanism.

7. The method of claim 6 wherein the score is calculated for each possible command identified after each character received in the text entry mechanism.

15 8. The method of claim 1 wherein identifying the possible commands is performed by a search component in a platform that supports a plurality of different applications.

9. The method of claim 1 wherein receiving a search request comprises:
receiving a speech input indicative of the search request.

20 10. The method of claim 1 and further comprising:

returning tutorial assistance in executing a selected command response to receiving a user selection of a possible command.

51039-28

11. A computer-readable medium having computer-executable instructions stored thereon that, when executed by a computer, cause the computer to implement the method of any one of claims 1 to 10.

12. A method of identifying commands in an application, comprising:

5 receiving a search request indicative of a desired task to be performed with the application;

searching a command store, that stores commands for the application, using the search request;

10 identifying as possible commands to perform the desired task, a subset of the commands in the command store based on the search request, the subset including a plurality of different commands for performing a plurality of different tasks in the application; and

returning the possible commands in a user actuable form such that one of the possible commands that corresponds to the desired task can be actuated by a user to execute the desired task in the application, wherein returning the possible commands comprises returning the possible commands in ranked order based on a score calculated for each possible command in the subset, the score being indicative of how likely it is that the possible command can be used to perform the desired task.

13. The method of claim 12 wherein identifying possible commands
20 comprises:

receiving a context input indicative of a context of the application and identifying the possible commands based on the context input.

14. The method of claim 12 wherein returning the possible commands in a user selectable form comprises:

51039-28

displaying a user actuatable interface element, for each possible command, that can be actuated by the user through a graphical user interface to perform the possible command.

15. The method of claim 12 wherein receiving a search request comprises:

5 displaying a text entry mechanism; and

receiving a text entry indicative of the search request.

16. The method of claim 15 wherein returning the possible commands in a user selectable form comprises:

10 displaying a user actuatable interface element, for each possible command, that can be actuated by the user through a graphical user interface to perform the possible command; and

displaying an associated command identifier, associated with each user actuatable interface element, wherein the command identifier can be entered in the text entry mechanism to perform the possible command.

15 17. The method of claim 15 wherein identifying possible commands and returning the possible commands are performed after each character received in the text entry mechanism.

18. The method of claim 17 wherein the score is calculated for each possible command identified after each character received in the text entry
20 mechanism.

19. The method of claim 12 wherein identifying the possible commands is performed by a search component in a platform that supports a plurality of different applications.

20. The method of claim 12 wherein receiving a search request comprises:

55450-8

receiving a speech input indicative of the search request.

21. The method of claim 12 and further comprising:

returning tutorial assistance in executing a selected command response to receiving a user selection of a possible command.

5 22. The method of claim 12 and further comprising:

conducting additional dialog to disambiguate a command is response to receiving a user selection of a possible command.

23. A system for locating a command to perform a task in any of a plurality of different applications, comprising:

10 a processor; and

a memory having processor-executable instructions stored thereon that, when executed by the processor, implement a searching component configured to receive a search request and a current context of an application in which the command is to be executed and to identify, from a plurality of commands, a plurality
15 of possible commands, comprising a subset of the plurality of commands, based on the search request and the context, the plurality of possible commands each actuable to perform a different task, and to return and display the plurality of possible commands in user selectable form, such that a selected one of the plurality of possible commands can be executed from the user selectable form to perform the
20 task in a given one of the plurality of different applications.

24. The system of claim 23, wherein the instructions, when executed, further implement:

a platform supporting the plurality of different applications, the platform including the searching component for conducting searching for the plurality of
25 different applications.

55450-8

25. The system of claim 24 wherein the processor, the memory including the instructions that implement the platform, and the searching component reside on a client device.

26. The system of claim 23 wherein the plurality of different applications
5 reside on a client device and further comprising:

a server connectable to the client device, the server including the processor, and the memory including the instructions that implement the searching component.

27. The system of claim 23 and further comprising:

10 a server including: one or more of the different applications; the processor; and the memory including the instructions that implement the searching component,

the one or more applications being accessible by one or more client devices.

15 28. The system of claim 23 wherein the searching component is accessible by a mobile device.

29. The system of claim 23 and further comprising:

memory comprising a command data store, searchable by the searching component and storing a command records indicative of commands for the
20 plurality of different applications.

30. A computer-readable medium having computer-executable instructions stored thereon which, when executed by a computer, cause the computer to perform steps of:

receiving a command search request;

55450-8

receiving context information indicative of a current context of an application for which a command is sought;

searching a data store to identify a plurality of possible commands based on the command search request and the context information, the possible
5 commands identified changing with changes in the context information; and

returning a list of the plurality of possible commands, each actuable to perform a different task, as user actuable graphical user interface elements, ordered according to a relationship between the possible commands and the search request and the context information.

10 31. A computer-readable medium having computer-executable instructions stored thereon that, when executed by a computer, cause the computer to implement the method of any one of claims 12 to 22.

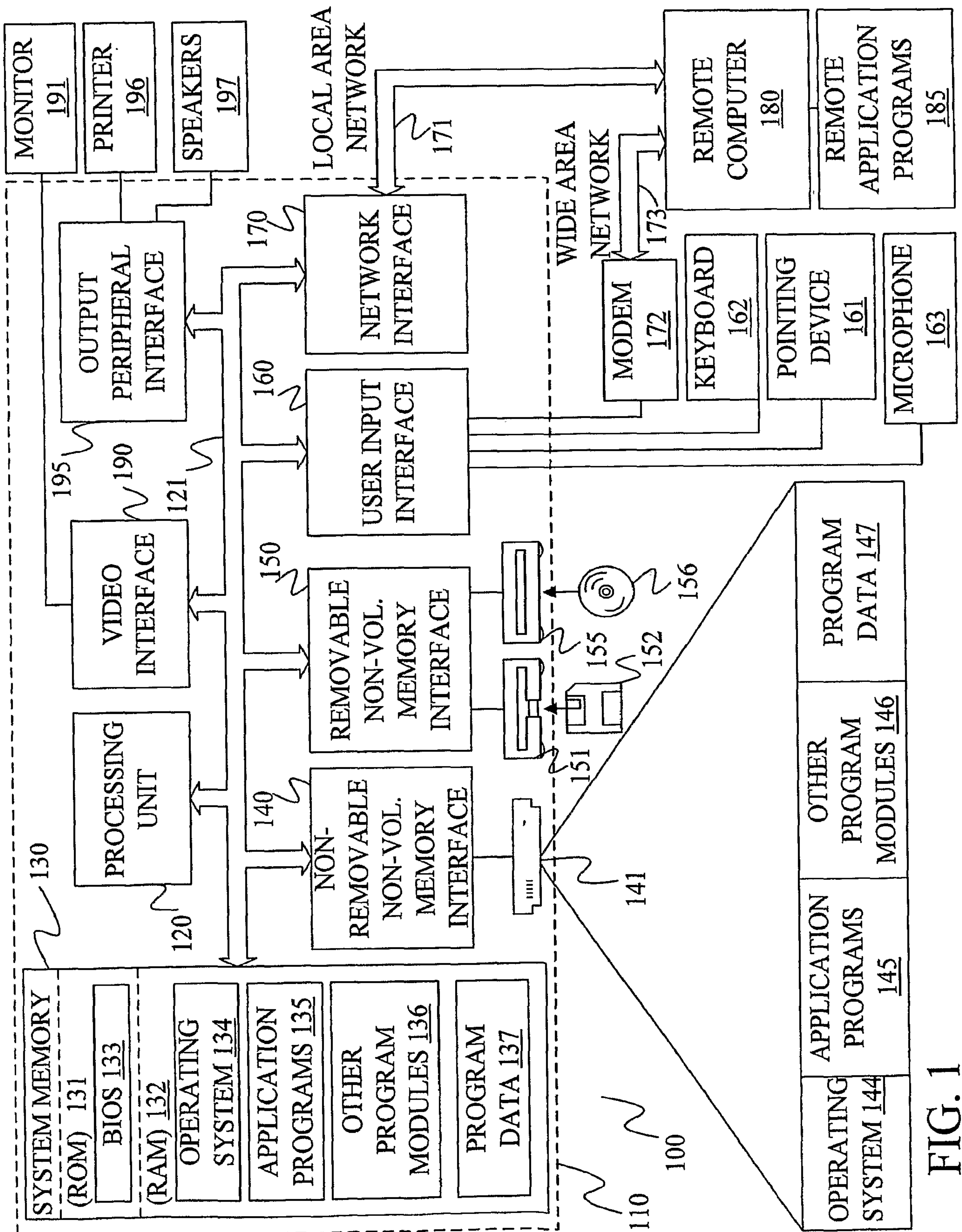


FIG. 1

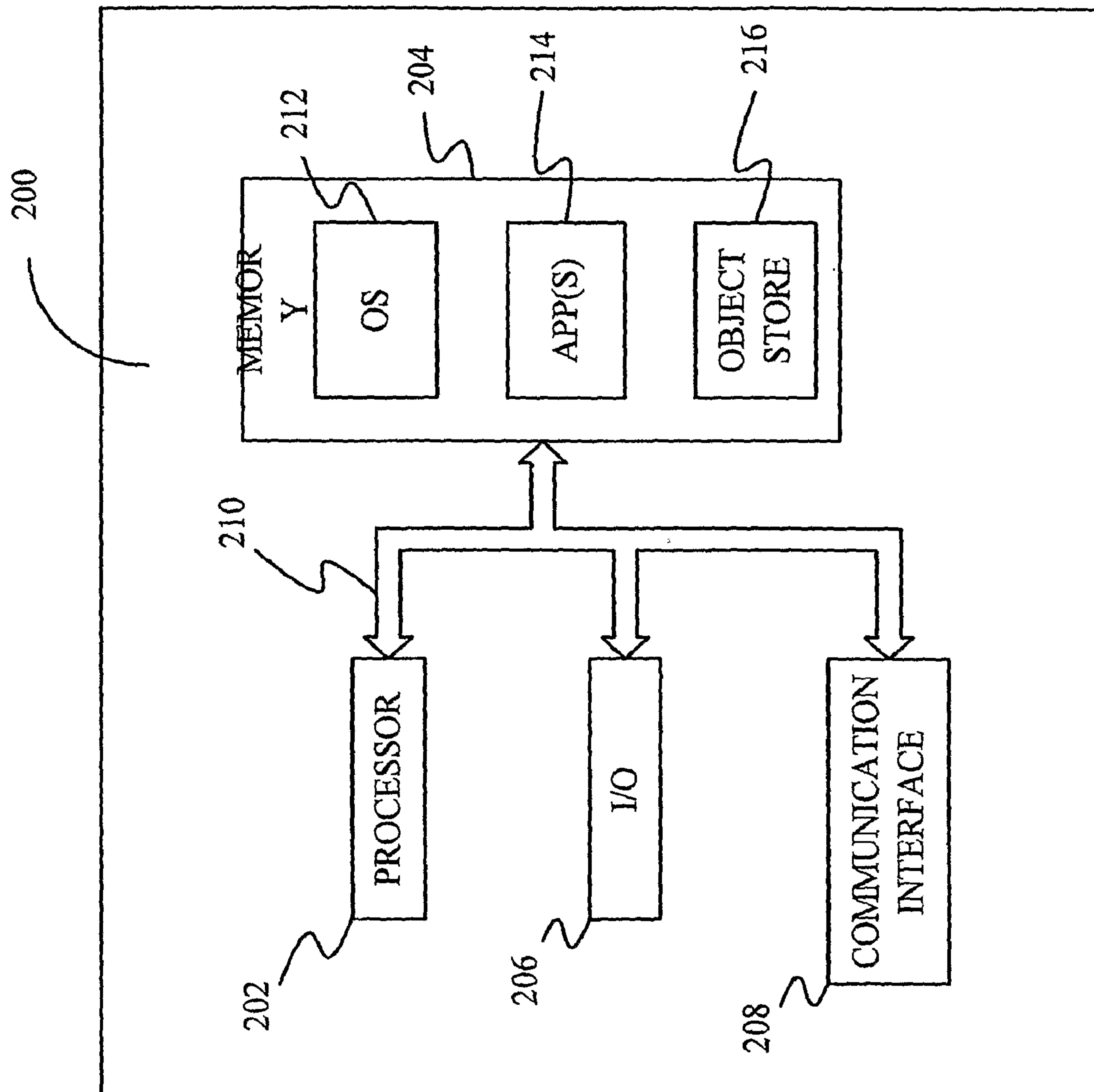
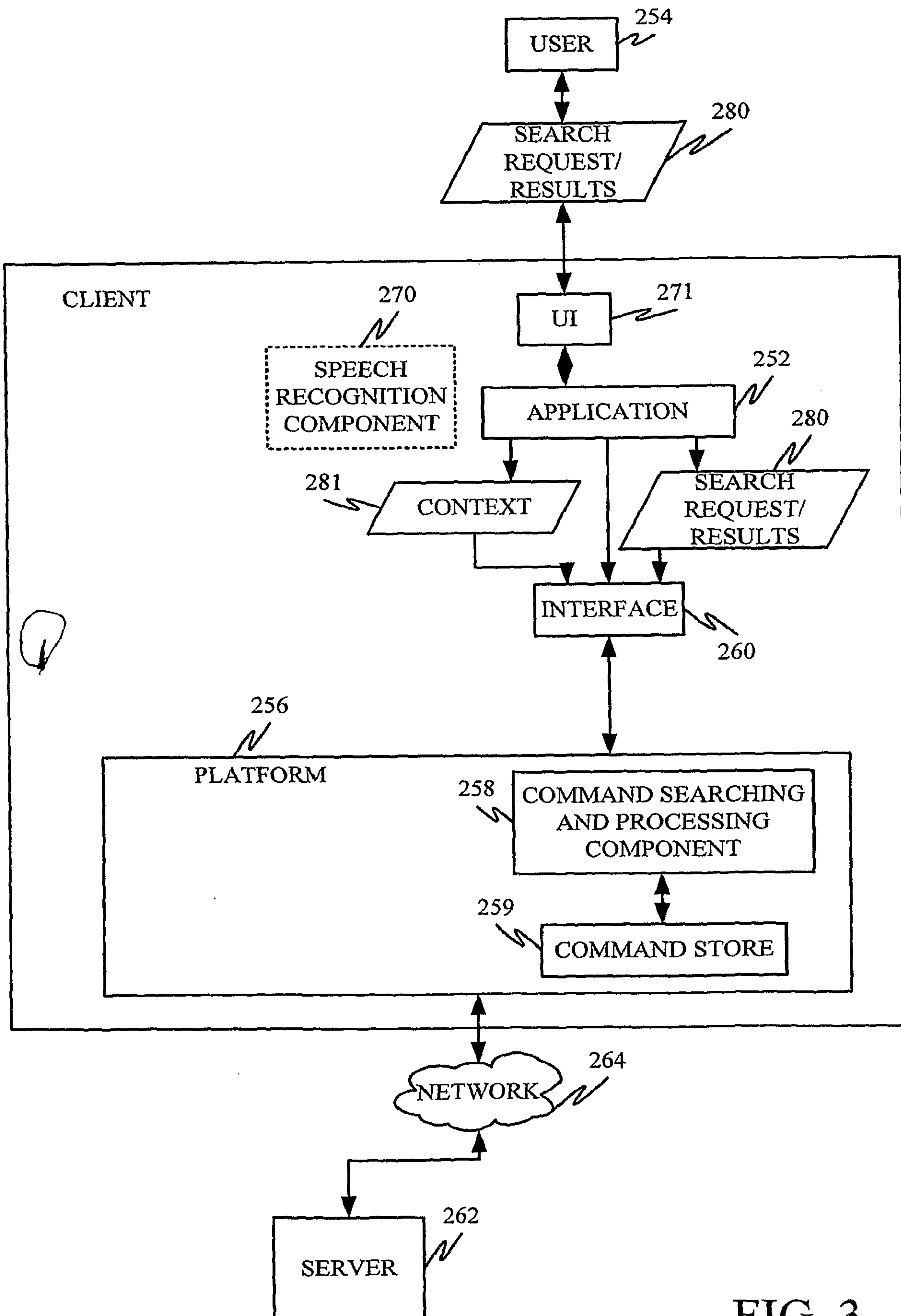
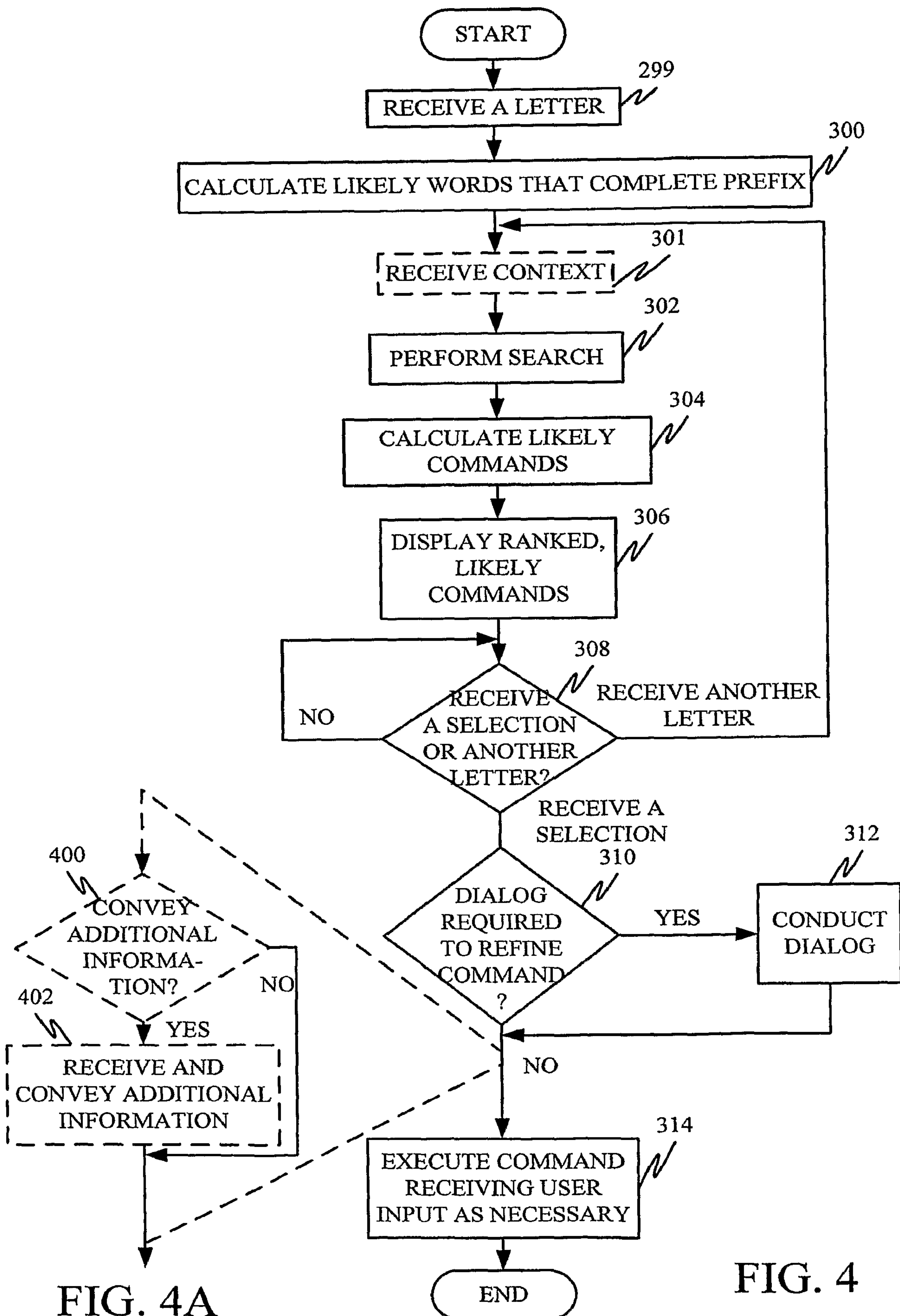


FIG. 2





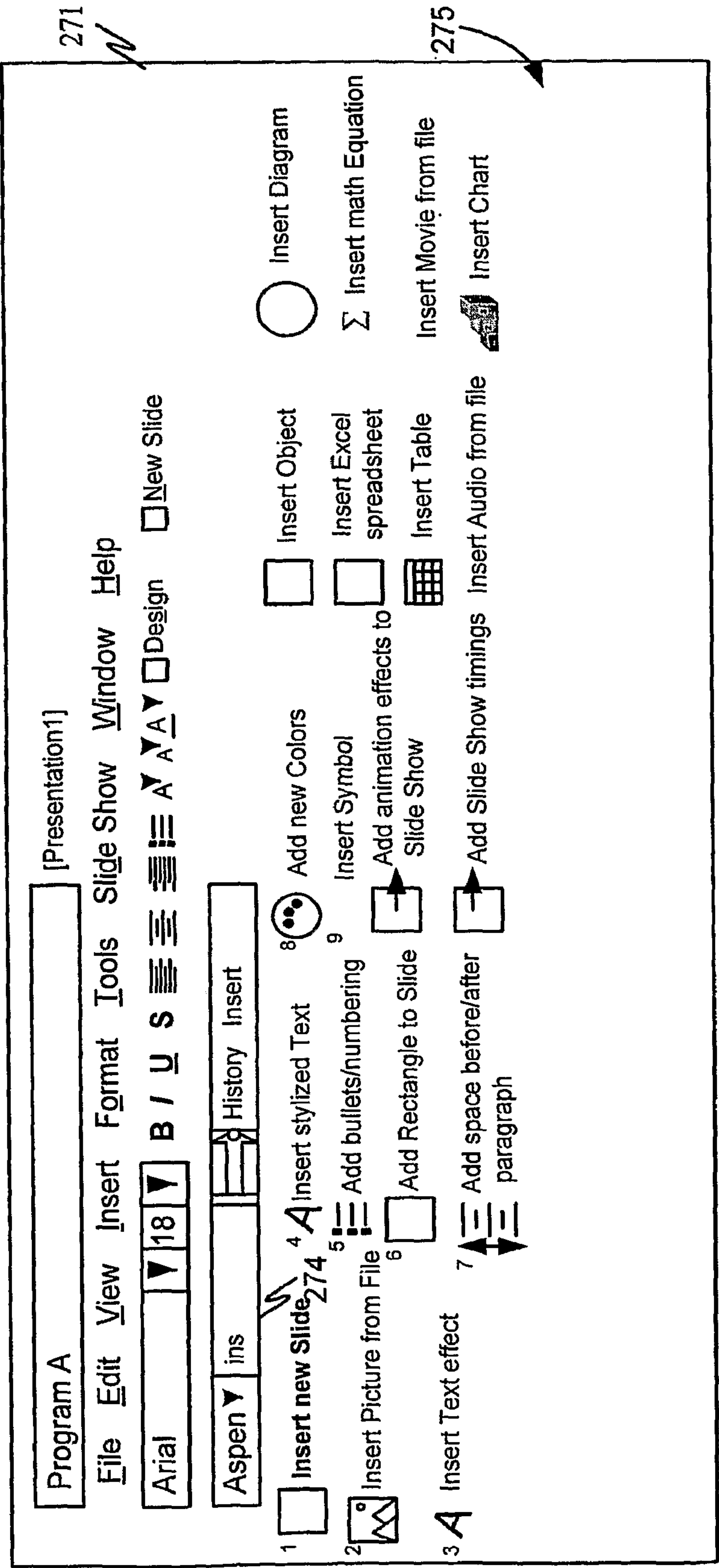


FIG. 5

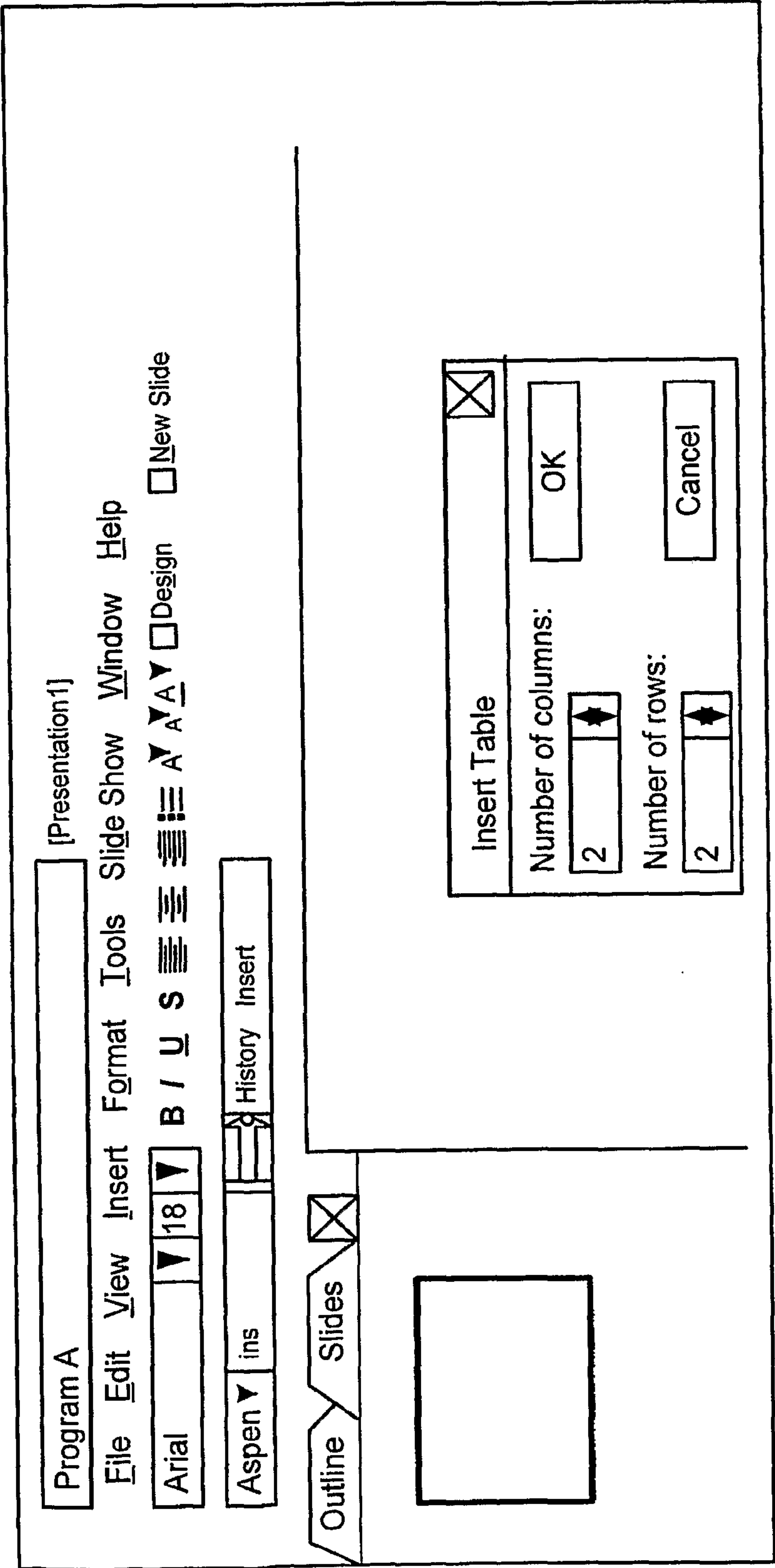


FIG. 6

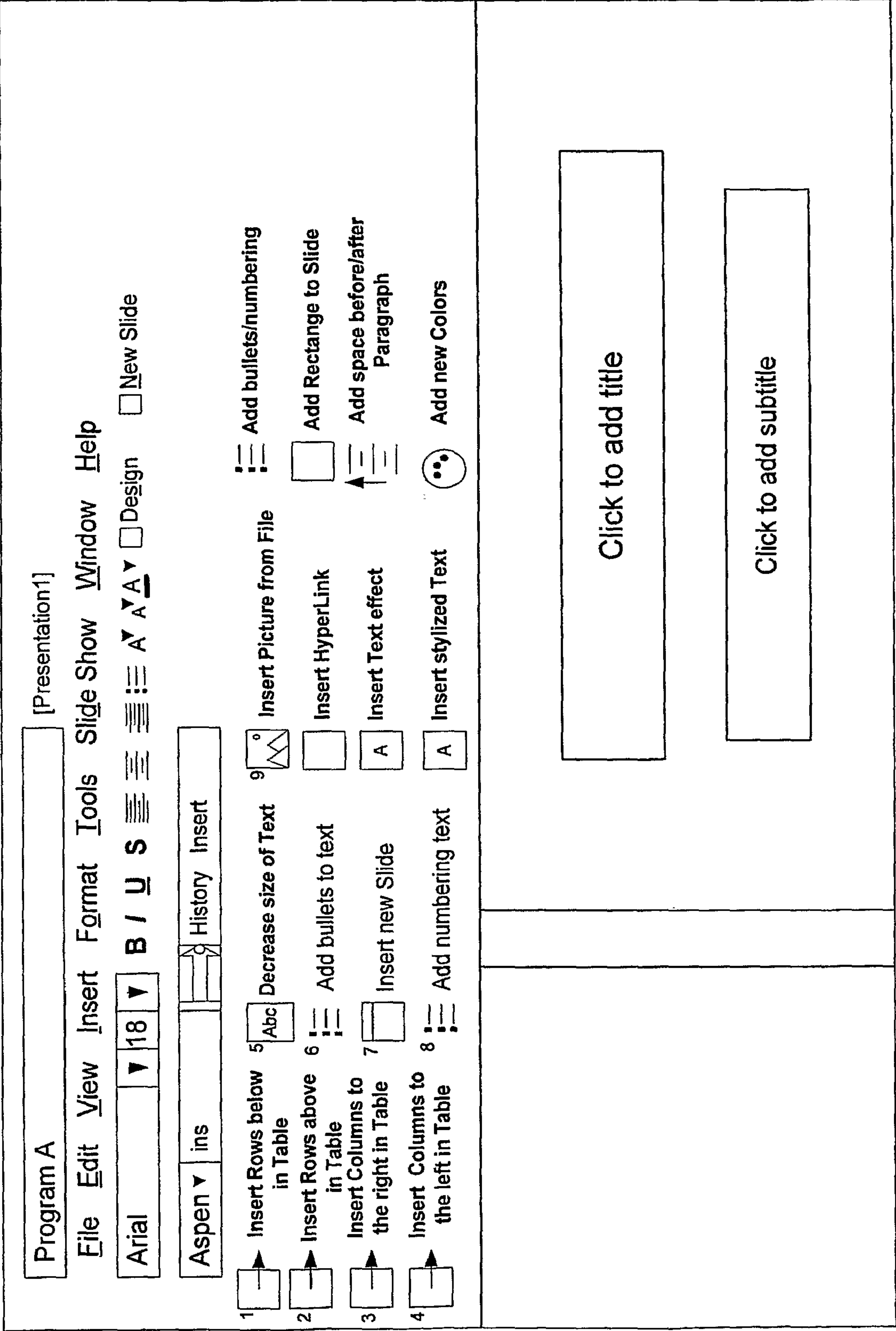


FIG. 7

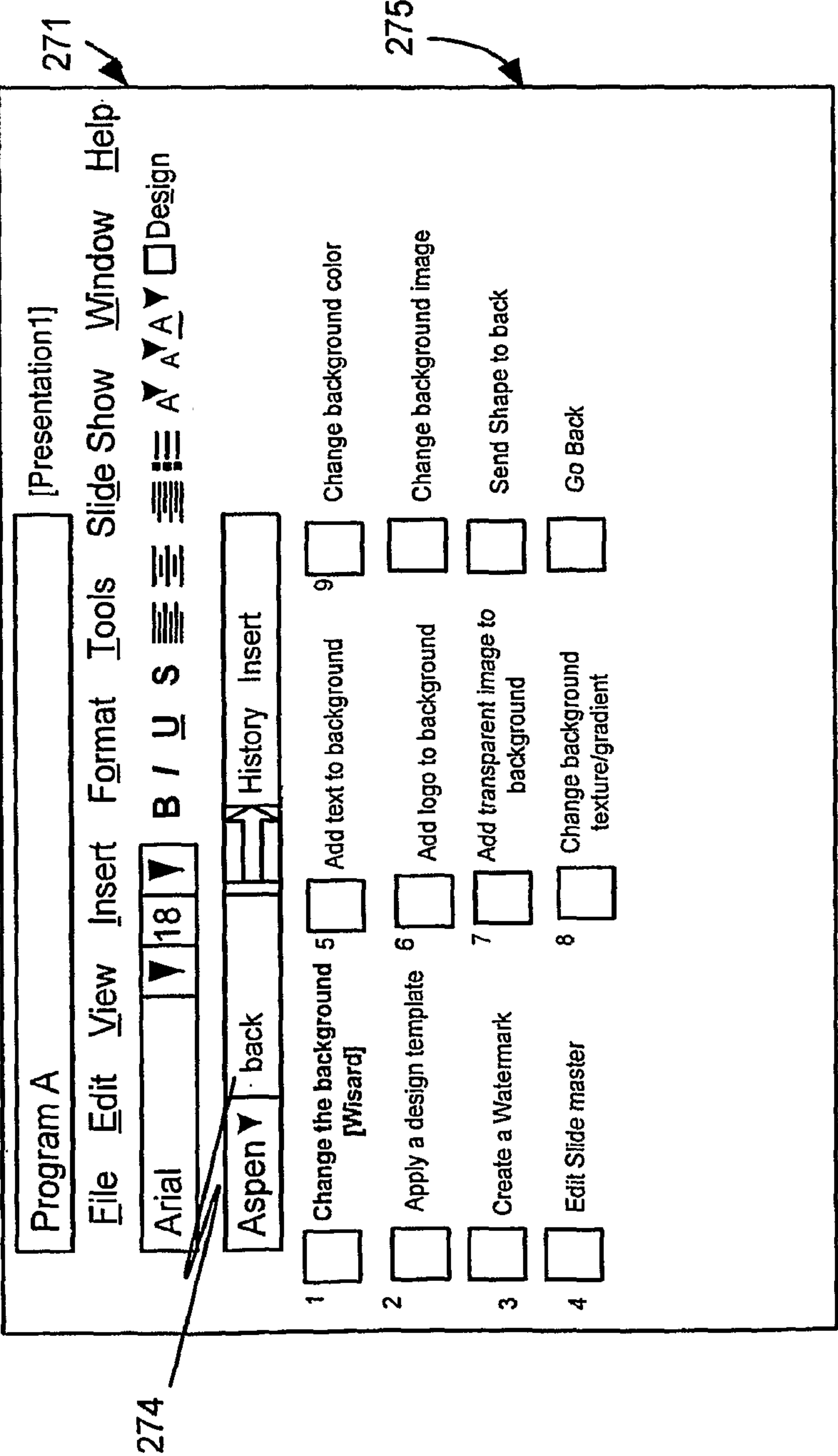
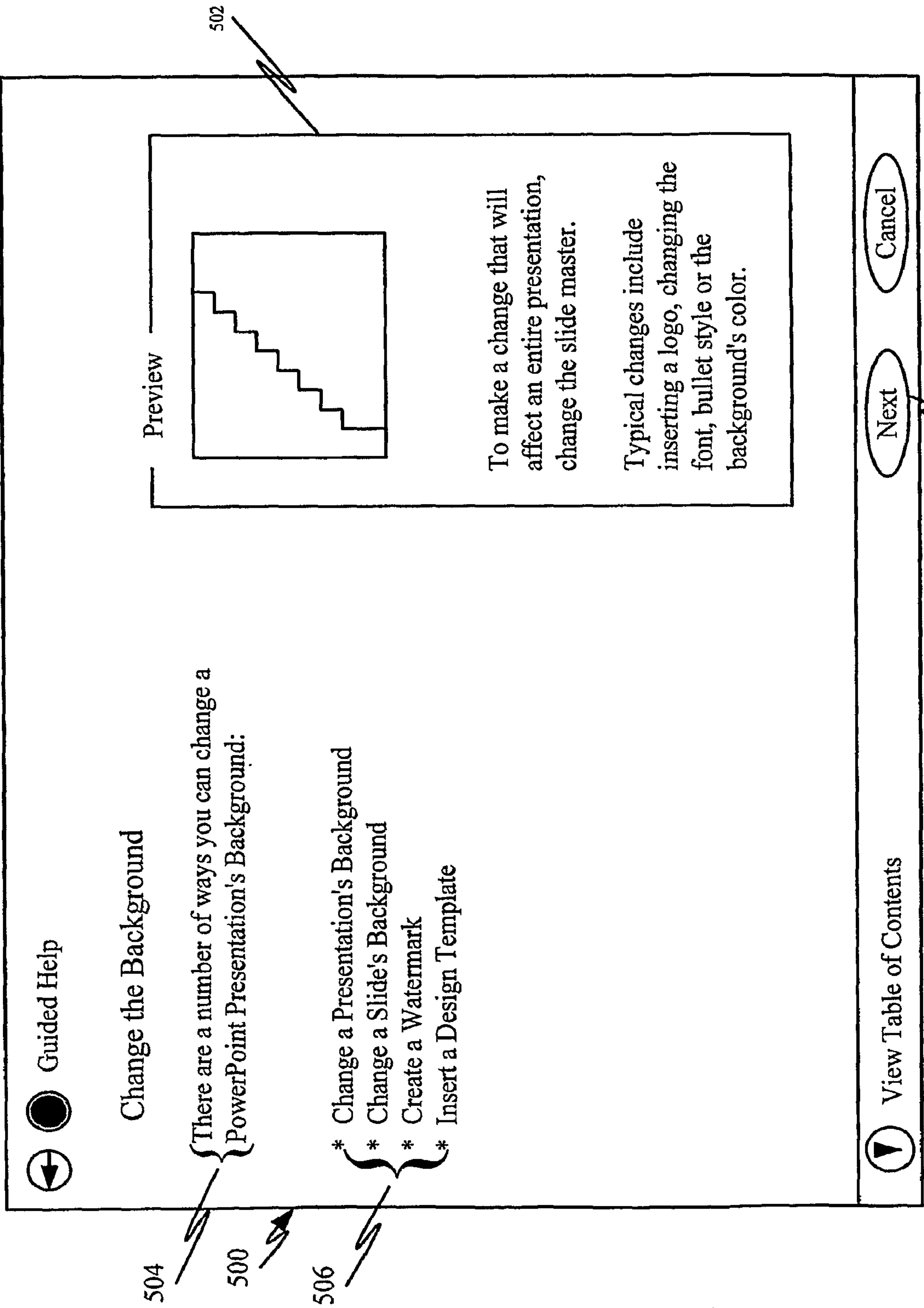


FIG. 8



271
N

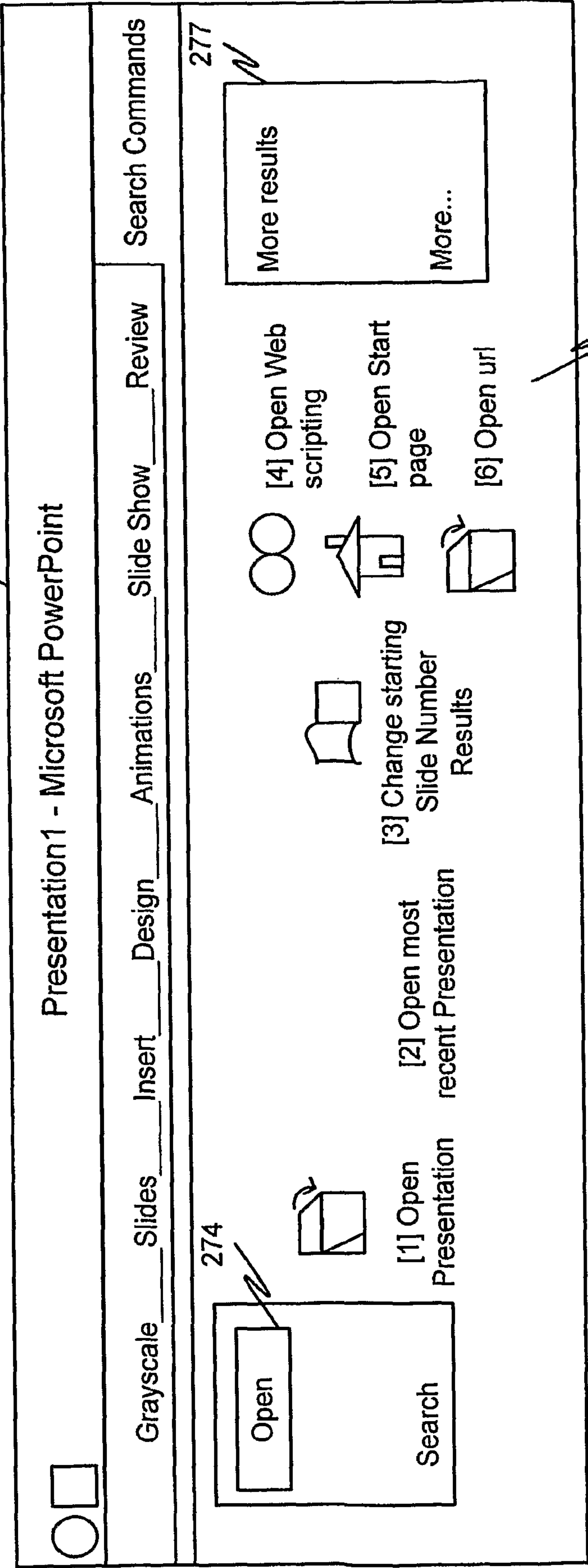


FIG. 10A

271

N

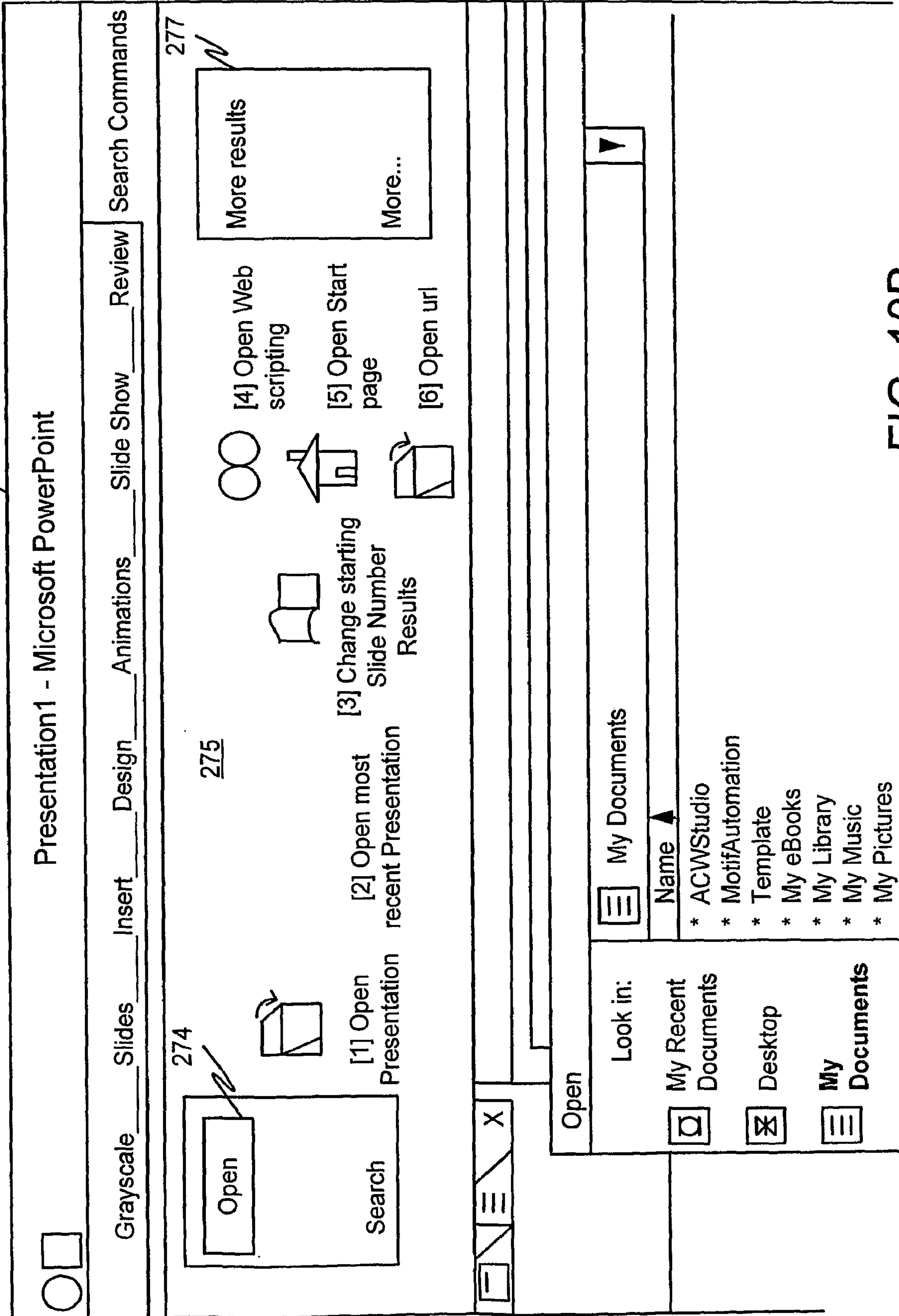


FIG. 10B

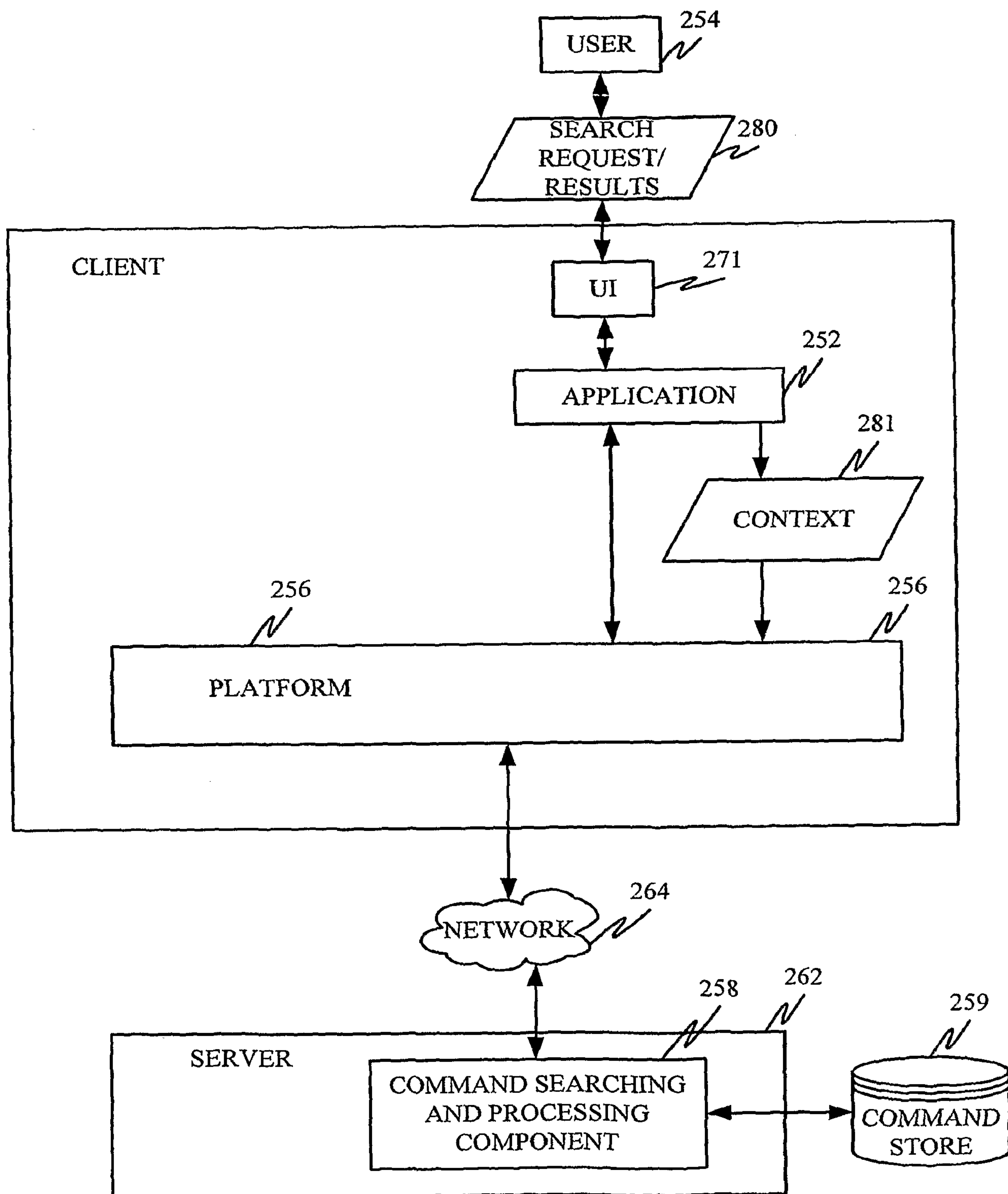


FIG. 11

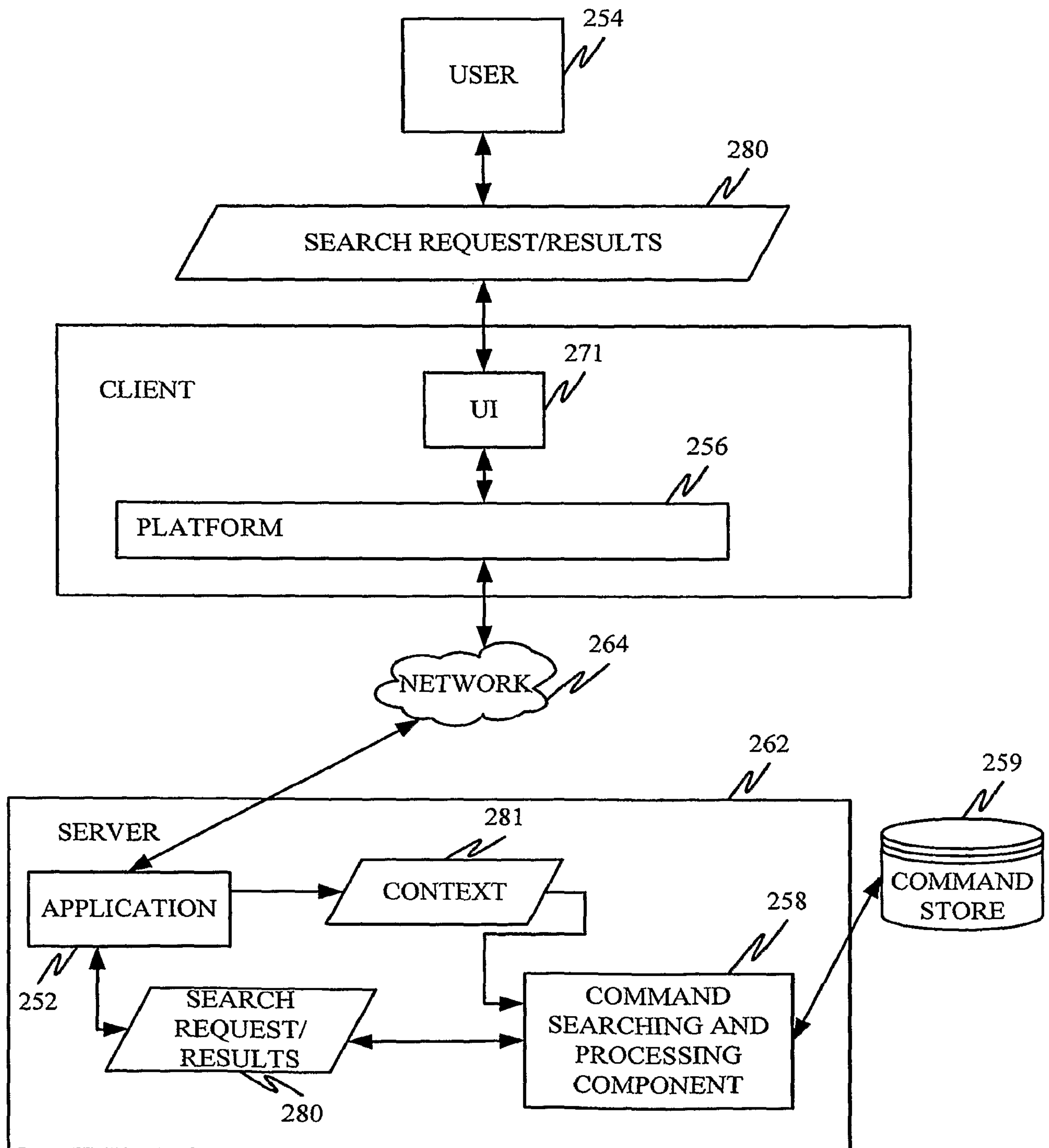


FIG. 12

