A system and method for proactively searching a plurality of objects in a graphical user interface (432), wherein each object, such as a window, icon, desktop, or other information container, results from an executing process in a multi-tasking environment. Upon a system encounter of an object that contains the search criteria, the system notifies the user both visually (402) and audibly (424).
FOR THE PURPOSES OF INFORMATION ONLY

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

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

Background of the Invention

Field of the Invention

This invention relates generally to automatic search techniques, and more specifically, to searching multiple software applications executing within a multitasking operating system.

Related Art

The growth and popularity of global computerized information systems within the last few years is staggering. What at one time required a person to travel to a library to research and seek information, now requires as little effort as "rolling out of bed" to a computer. The underlying foundation for global information systems is Internet technology. Internet technology is a way of linking networks together so that all kinds of computers can work together and exchange data, regardless of how similar or dissimilar the computers are from an electronic standpoint.

A communication system based on the Internet is the World Wide Web (WWW). In fact, it was in 1993 that the WWW exploded in popularity, such that now the Web is considered the fastest-growing communications system that history has ever witnessed.

The WWW is a geographically distributed pool of information, so that users separated by short or long distances can make information available to others. The WWW combines two technologies -- hypermedia and Internet. Users use the WWW to access multimedia documents for nonsequential access to multimedia information and use graphical user interface (GUI) client programs capable of searching the WWW for specific information.

Amid the apparent wealth of online resources and information, users still have problems finding the right kind of relevant information. Current search services do not search every piece of information that is visually encountered by a
user. If they did, a user would wait almost infinitely in time for a complete candidate list from a resource such as the Internet. Internet search services provide searches on indexes, abstracts, or specific file types which highlight data satisfying the search criteria. The user may then explore further a reference address that corresponds to a specific index or abstract of interest.

Furthermore, a user often wants to search across multiple software applications executing on a computer. One problem a user faces is how to specify the search criteria within each application's search function. Users must learn and remember how to use a variety of metaphors, user interfaces, and searching techniques for each application's search function and class of information. There is not one type of search interface that suits all users or applications.

A user, depending on the circumstances of the specific information needed, must use different kinds of searching techniques and different search interfaces. Applications, however, typically provide only one type of search function. Therefore, the available search function may be inadequate for the user's needs. If the search function is inappropriate for the user or for the current circumstance, the search task fails, or will be less than successful, in finding the needed information. Furthermore, an application may not provide any search capability.

Therefore, a method and system is needed for facilitating searches of information across multiple software applications executing within a multi-tasking operating system, thereby ensuring that all information of interest is found in a timely manner. A method and system is further needed for searching multiple software applications that have no, or inadequate, search capabilities, thereby globalizing a search function to any software application that a user executes.

Summary of the Invention

The present invention solves the problems described above by providing a method and system for a proactive search capability to multiple software applications executing within a multi-tasking operating system. The search capability extends to any current or forthcoming representation, or user interface object (object), of an
application. For example, the present invention searches all windows, icons, desktops, or other information containers that may occur within a graphical user interface of a computer system.

A user configures the search criteria prior to initiating the search. After configuring the search criteria, the user simply navigates through any application, in any manner. The present invention automatically searches each object invoked by the operating system according to the search criteria. Upon a system encounter of an object that contains the search criteria, the present invention notifies the user. User notification may comprise a configurable audible tone, such as a beep, a voice annotated text string, or a recording. Furthermore, objects containing the search criteria have at least one visible attribute modified during the time that the object contains the search criteria. Attribute modification may include having a border thickened, bolding text, etc. Object attributes are automatically restored as soon as the object no longer contains the search criteria.

In a multi-tasking operating system, the present invention allows a user to invoke a plurality of arbitrary applications which can produce a variety of user interface objects during their processing. Some of the applications may navigate in an automated manner. The present invention identifies search criteria as it becomes detected in forthcoming objects associated with the arbitrary applications.

The user may conveniently transpose to an object that contains the search criteria at any time. The present invention provides a perusal list hot-key for traversing all of the objects that contain the search criteria in a round-robin sequence. A user may also make a perusal list request to view a specific object in the perusal list. The present invention also provides a jump-from hot-key for returning the user back to the object with which the user was previously working at the time of initiating the view perusal list hot-key.

The present invention also provides a configurable mode for obtaining search results. A preempt mode automatically interrupts a user as soon as the present invention detects that an object contains the search criteria. In a preempt mode, the present invention displays the object that contains the search criteria to the user and focuses it to the user as soon as the search criteria has been detected. In a passive
mode, a user continues normal operating system and/or application use in an uninterrupted fashion while detecting search criteria is audibly announced to the user. The user only traverses the results of a search when he requests such an action.

The present invention allows a user to concentrate on fast navigation of applications while the operating system automatically searches objects displayed on the graphical user interface. The present invention detects search criteria in objects that may not be viewable in the object at the time it is first exposed. The operating system searches objects at the system level, not at the visual level, wherein the visual level is the portion of an object displayed to the user. For example, an object may comprise 500K bytes of searchable text in a scrollable window at the system level, but comprise only 200 visual words of text in that scrollable window displayed to the user. Therefore, the present invention finds search criteria which may have been missed by a user because of not navigating to an appropriate destination within the object, or because the user simply overlooked the content of the object.

Additional features of this invention will become apparent from the following detailed description of the best mode for carrying out the invention and from the appended claims.

**Brief Description of the Figures**

The present invention is described with reference to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements. Additionally, the left-most digit(s) of a reference number identifies the drawing in which the reference number first appears.

FIG. 1 is a block diagram illustrating an environment in which the present invention operates;

FIG. 2 is a block diagram of an exemplary computer system of the present invention;
FIG. 3A is a level 0 data flow diagram illustrating the Proactive Searcher external interfaces;

FIG. 3B is a level 1 data flow diagram illustrating the Proactive Searcher components;

FIG. 3C is a control flow diagram representing the operation of configuring the present invention;

FIG. 3D is a control flow diagram representing the operation of searching objects for search criteria;

FIG. 4 is a control flow diagram representing the operation of configuring search criteria and other configuration parameters;

FIG. 5 is a control flow diagram representing the operation of building a perusal list upon completion of configuring search criteria and other configuration parameters;

FIG. 6 is a control flow diagram representing the operation of visually focusing an object to a user;

FIG. 7 is a control flow diagram representing the operation of viewing the perusal list;

FIG. 8 is a control flow diagram representing the operation for exposing the user's original object;

FIG. 9 is a control flow diagram representing the operation for creation of an object in the GUI;
FIG. 10 is a control flow diagram representing the operation for the change of an object in the GUI; 

FIG. 11 is a control flow diagram representing the operation of removing an object from the perusal list; 

FIG. 12 is a control flow diagram representing the operation of adding an object to the perusal list; and 

FIG. 13 is a control flow diagram representing the operation for the removal of an object in the GUI.

Detailed Description of the Preferred Embodiments

1. Overview

The present invention is a proactive searcher that automatically searches multiple software applications executing within a multi-tasking operating system. More specifically, the proactive searcher searches all objects displayed in a graphical user interface. Objects include icons, windows (and child-windows and child-regions thereon), desktops, or any other informational container that displays information to a user. Even buttons and control areas of windows are searched.

FIG. 1 is a block diagram illustrating an environment in which the proactive searcher operates. The display device 102 represents a computer screen which employs a graphical user interface. Software applications executing within the computer system can create multiple objects each with varying functions.

A presentation manager maintains a list of windows, such as Window A 104(a), Window B 104(b), and Window C 104(c), icons, such as Icon 1 106(a) and Icon 2 106(b), or other window-type objects created by the executing applications. The presentation manager associates each object with the application that created it.

When an application terminates, the presentation manager destroys the objects that
the application created if the application itself has not already destroyed them. Windows 104 may contain a variety of child-windows or child regions thereon, for example scroll windows, buttons, group areas, etc. All items which may appear on a window are searchable.

2. Preferred Embodiment

The chosen embodiment of the present invention is computer software executing within a computer system. FIG. 2 shows an exemplary computer system. The computer system 202 includes one or more processors, such as a processor 204. The processor 204 is connected to a communication bus 206.

The computer system 202 also includes a main memory 208, preferably random access memory (RAM), and a secondary memory 210. The secondary memory 210 includes, for example, a hard disk drive 212 and/or a removable storage drive 214, representing a floppy disk drive, a magnetic tape drive, a compact disk drive, a compact disk drive, a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as EPROM, or PROM), etc. which is read by and written to by a removable storage unit 216. Removable storage unit 216, also called a program storage device or a computer program product, represents a floppy disk, magnetic tape, compact disk, etc. As will be appreciated, the removable storage unit 216 includes a computer usable storage medium having stored therein computer software and/or data. The removable storage drive 214 reads from and/or writes to a removable storage unit 116 in a well known manner.

The computer system 202 may also include other similar means for allowing computer programs or other instructions to be loaded. Such means can include, for example, a communications interface 218. Communications interface 218 allows software and data to be transferred between computer system 202 and external devices. Examples of communications interface 218 can include a modem, a network interface (such as an Ethernet card), a communications port, etc. Software and data transferred via communications interface 218 are in the form of signals
which can be electronic, electromagnetic, optical or other signals capable of being received by communications interface 218.

In this document, the term "computer program product" is used to generally refer to removable storage unit 216, a hard disk installed in hard disk drive 212, and signals transferred via communications interface 218. These computer program products are means for providing software to a computer system 202.

In an embodiment where the invention is implemented using software, the software may be stored in main memory 208, or in a computer program product and loaded into computer system 202 using removable storage drive 214, hard disk drive 212, or communications interface 218. The software, when executed by the processor 204, causes the processor 204 to perform the functions of the invention as described herein.

In another embodiment, the invention is implemented primarily in hardware using, for example, a hardware state machine. Implementation of the hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant arts.

The preferred embodiment of the present invention is directed to a computer system 202 executing a multi-tasking operating system such as OS/2 or UNIX. A multi-tasking operating system allows multiple applications, or processes, to execute concurrently. More specifically, the preferred embodiment of the present invention is implemented in software coupled to a presentation manager of a multi-tasking operating system. After reviewing the disclosure provided below, it would be readily apparent to one skilled in the relevant art to make the necessary modifications to a presentation manager. In an alternate embodiment, the present invention can be coupled to a presentation manager of a software application. The preferred embodiment is described in these terms for convenience purpose only. Other operating systems and couplings could alternatively be used.

The preferred embodiment of the present invention is also directed to a computer system 202 including a display device 102 and one or more input peripherals 220. As described above, the display device 102 represents a computer screen or monitor on which a graphical user interface may be displayed. The input
peripherals 220 include, for example, a keyboard, a mouse, a light pen, a pressure-sensitive screen, etc. which provide a user with the capability of entering input to the computer system 202.

3. Data Flow for the Proactive Searcher

FIG. 3A is a level 0 data flow diagram illustrating the external interfaces of the proactive searcher. A data flow diagram represents how data is used among the components of a computer software application. A level 0 data flow diagram illustrates the highest level of data interfaces between a software application and external sources. Therefore, in the level 0 data flow diagram of FIG. 3A, the computer software application being depicted is the proactive searcher 302 which is represented by the rounded-rectangle (or bubble). The graphical user interface 304 represents the external source of data that interfaces with the proactive searcher 302.

In a data flow diagram, the direction of an arrow connecting two objects represents the directional flow of data. For example, the graphical user interface 304 sends user input to the proactive searcher 302. User input includes key strokes, or key stroke combinations, and configuration setup data such as search criteria, object attributes, and user notification settings.

In response to user input, the proactive searcher 302 returns display objects to the graphical user interface 304. More specifically, the proactive searcher 302 sends an object, having specific object attributes such as highlights, bolding, etc., to the graphical user interface 304 for display. The object sent to the graphical user interface 304 is one which the proactive searcher 302 has determined should be focused, or needs to be brought to the user’s attention.

FIG. 3B is a level 1 data flow diagram illustrating the components of the proactive searcher 302 and their relationship to the external interface (as shown in FIG. 3A) and to each other. The components of the proactive searcher 302 include: the configuration handler 306, object handler 308, perusal list handler 310, and the semaphore handler 312. It should be understood that the invention is
described with reference to these components for convenience purpose only. Other software components could alternatively be used as would be readily apparent to one skilled in the art.

The configuration handler 306 receives user input via the graphical user interface 304 (as shown in FIG. 3A). Based on the user input, the configuration handler 306 sets up various configuration parameters for the proactive searcher 302. In this circumstance, the user input comprises a keystroke, combination of keystrokes, mouse selection, or the like, to invoke the configuration process of the proactive searcher 302. After initiating the configuration process, the user inputs various configuration parameters, including: object attributes, search criteria, search mode, and user notification. These user inputs are described in greater detail below.

The object handler 308 performs all of the functions associated with an object displayed within a graphical user interface 304 on a display device 102. Object functions may include, but are not limited to, creating an object, deleting an object, modifying an object, searching an object for search criteria, and displaying an object according to object attributes. The object handler 308 receives user input via the graphical user interface. In this circumstance, the user input comprises a keystroke, combination of keystrokes, mouse selection, or the like, to invoke an object function. Subsequent processing may also automatically invoke an object function. After a user initiates an object function, the object handler 308 executes the appropriate function and if required, displays an object to the user via the graphical user interface 304. The object functions are described in greater detail below.

The object handler 308 also invokes the configuration handler 306. This control flow is denoted by the dashed arrow drawn from the object handler 308 to the configuration handler 306. The object handler 308 accesses the configuration handler 306 to receive the configuration parameters in return. The object handler 308 uses the configuration parameters to perform its object functions.

The perusal list handler 310 performs all of the functions associated with a perusal list, which is a list of objects displayed within a graphical user interface 304
on a display device 102 such that each object in the list satisfies the currently configured search criteria. Perusal list functions may include, but are not limited to, creating a perusal list, adding an object to a perusal list, deleting an object from a perusal list, and counting the objects in a perusal list. The perusal list functions are described in greater detail below. In an alternate, Internet embodiment, the perusal list handler 310 saves the Internet, or World Wide Web, address associated with an object. Therefore, when the user wants to view a specific object from the perusal list, the perusal list handler 310 would automatically return the object's corresponding Internet address for optional automatic transportation to the address. For example, an entry in a perusal list would be an Internet address such as a WWW homepage or a context therein.

The object handler 308 invokes the functions of the perusal list handler 310. This control flow is denoted by the dashed arrow between the object handler 308 and the perusal list handler 310. The object handler 308 accesses the perusal list handler 310 to receive object data. Object data is information on a specific object in the perusal list. The object handler 308 uses the object data to perform its object functions.

The Semaphore handler 312 manages access to a single semaphore, or flag, thereby regulating access to shared resources. The semaphore synchronizes multiple processes in a multi-tasking operating system such that only one process has access to the shared resources at a time. Therefore, if a first process "locks" the semaphore, a second process cannot access the shared resources until the first process "unlocks" the semaphore, thereby freeing up the shared resource for another process to access. Semaphore functionality is preferably provided by the operating system. Semaphore control is well known to those skilled in the relevant art.

In the preferred embodiment of the proactive searcher 302, the shared resources protected by the semaphore handler 312 includes several variables, such as the perusal list, INDEX variable, and JUMP_FROM variable. As described above, the perusal list contains a list of all of the objects which satisfy the currently configured search criteria. In addition to the perusal list, an implicit count of the
number of objects contained in the perusal list is maintained to ensure that objects are added and deleted properly from the perusal list. Each object entry in the perusal list comprises at least two fields. One field contains the thirty-two (32) bit handle, or identifier, of the object. The presentation manager assigns the handle at the time it creates the object. A second field contains an original value of an object attribute. The object attribute is the one chosen to indicate that the object contains the search criteria. Therefore, this field is used to restore the object attribute back to its original value when the object no longer contains the search criteria. The configuration handler 306 allows a user to specify one or more object attributes which should be automatically modified upon containing a search criteria. There will be N+1 fields in a perusal list entry when a user configures N attributes to be automatically detected.

Object entries may contain additional information. For example, if more than one object attribute is affected by the existence of search criteria, the object entries must contain multiple fields. Each affected attribute requires a separate field. Furthermore, if automatic highlighting occurs within an object upon the detection of search criteria, the object entry must contain the offset to the text string with the text string length and the graphical position information. Therefore, a field in a perusal list entry may comprise multiple sub-fields.

The INDEX variable always points to the next perusal list entry to which the user may view in a round-robin examination of the objects contained in the perusal list. A user engages a round-robin examination by repeatedly invoking the perusal list hot-key. The JUMP_FROM variable contains the object handle of the object which the user left just prior to invoking the perusal list hot-key. This variable provides the user with a convenient mechanism for returning back to what the user was doing just prior to examining search results. In addition, a user may make a perusal list request to view a specific object in the perusal list.
4. Control Flow Overview for the Proactive Searcher

FIG. 3C is a control flow diagram representing the operation of configuring the present invention. Control begins at step 316 and immediately proceeds to step 318. In step 318, the configuration handler 306 (proactive searcher 302) interacts with a user for configuring the search criteria and other configuration parameters. Step 318 is described in greater detail below. Continuing to step 320, the configuration handler 306 (proactive searcher 302) exits its processing.

FIG. 3D is a control flow diagram representing the operation of processing objects. Control begins at step 324 by way of newly configuring search criteria (as described below in FIG. 4), creating an object in the system (as described below in FIG. 9), modifying an object in the system (as described below in FIG. 10), or removing an object from the system (as described below in FIG. 13). Once step 324 is invoked, processing immediately proceeds to step 326. In step 326, the object handler 308 (proactive searcher 302) traverses the object(s) displayed on the graphical user interface 304 and processes them appropriately. Processing includes searching the created or modified object(s) for the user configured search criteria and notifying the user of any occurrences of search criteria within an object, as well as searching the destroyed object(s) for the user configured search criteria in order to update the perusal list. Step 326 is described in greater detail below with reference to FIGs. 4, 9, 10, and 13. Continuing to step 328, the object handler 308 (proactive searcher 302) exits its processing.

5. Control Flow for the Configuration Handler

FIG. 4 is a control flow diagram representing step 318 in FIG. 3C and the configuring of search criteria and configuration parameters. Control begins at step 400 and immediately continues to step 402. In step 402, the configuration handler 306 displays to the user the current configuration criteria, including the settings for current search criteria (discussed below), object attributes set in step 407 (discussed below), search mode set in step 422 (discussed below), and a reference to the
audible notification set in step 426 (discussed below). Continuing to step 404, the configuration handler 306 waits for user input. The user’s input is a setup command that allows the user to modify a specific configuration parameter. Once the configuration handler 306 detects a user input, the configuration handler continues to step 406.

In step 406, the configuration handler 306 determines whether the user chooses to set object attributes. More specifically, this command allows a user to pick one or more attributes of an object that will be changed when the proactive searcher 302 determines that the object contains the search criteria. If the configuration handler determines that the user input is the set attribute command, the configuration manager proceeds to step 407. In step 407, the configuration handler 306 sets the objects’ attributes as selected by the user. Such object attributes may include, but are not limited to, highlighting the search criteria in the object, bolding the object, making the border of the object thicker, having the object flash, or changing the color of the object.

Continuing to step 428, the configuration handler 306 saves the current configuration criteria as changed by the user. Afterwhich, the configuration handler 306 returns to step 402 and displays the changed configuration parameters to the user. At step 402, the configuration handler repeats the process described above and waits for the next user input.

Referring again to step 406, if the configuration handler 306 determines that the user input was not the set attribute command, the configuration handler 306 proceeds to step 408. In step 408, the configuration handler 306 determines whether the user input is the add search criteria command. This command allows the user to add search criteria to the current search criteria. If the configuration handler 306 determines that the user entered the add search criteria command, the configuration handler 306 proceeds to step 410. In step 410, the configuration handler 306 adds the new search criteria entered by the user. Continuing to step 428, the configuration handler 306 saves the new search criteria and current configuration parameters as changed by the user. Afterwhich, the configuration handler 306 returns to step 402 and reenters the processing loop described above.
The user may configure a plurality of search criteria specifications through step 410.

Referring again to step 408, if the configuration handler 306 determines that the user input was not the add search criteria command, the configuration handler 306 proceeds to step 412. In step 412, the configuration handler 306 determines whether the user input is the delete search criteria command. This command allows a user to delete search criteria from the current search criteria. If the configuration handler 306 determines that the user entered the delete search criteria command, the configuration handler 306 proceeds to step 414. In step 414, the configuration handler 306 deletes the criteria as specified by the user. After which, the configuration handler 306 returns to step 402 and reenters the processing loop described above.

Referring again to step 412, if the configuration handler 306 determines that the user input is not the delete search criteria command, the configuration handler 306 proceeds to step 416. In step 416, the configuration handler 306 determines whether the user input is the change search criteria command. This command allows a user to change the current search criteria. If the configuration handler 306 determines that the user entered the change search criteria command, the configuration handler 306 proceeds to step 418. In step 418, the configuration handler 306 changes the criteria as specified by the user. After which, the configuration handler 306 returns to step 402 and reenters the processing loop described above.

Referring again to step 416, if the configuration handler 306 determines that the user did not input the change search criteria command, the configuration handler 306 proceeds to step 420. In step 420, the configuration handler 306 determines whether the user input is the set mode command. This command allows a user to set the search mode as preemptive or passive. The preemptive mode authorizes the proactive searcher 302 to interrupt the user and display an object that contains the search criteria. The passive mode requires the proactive searcher 302
to wait for a user command to display the results, if any, of a search. If the configuration handler 306 determines that the user entered the set mode command, the configuration handler 306 proceeds to step 422. In step 422, the configuration handler 306 sets the search mode as specified by the user. Continuing to step 428, the configuration handler 306 saves the current configuration parameters as changed by the user. After which, the configuration handler 306 returns to step 402 and reenters the processing loop described above.

Referring again to step 420, if the configuration handler 306 determines that the user input is not the set mode command, the configuration handler 306 proceeds to step 424. In step 424, the configuration handler 306 determines whether the user input is the set audible notification command. This command allows a user to select how the proactive searcher 302 is to audibly notify him of the fact that one or more objects satisfy the search criteria. Notification may include, but is not limited to, an audible announcement such as a beep or recording, or a broadcast textual message. In the preferred embodiment, the user sets an audible notification. The use of an audible notification is for convenience purposes only. Alternate notifications can be used. If the configuration handler 306 determines in step 424 that the user entered the set audible notification command, the configuration handler 306 proceeds to step 426. In step 426, the configuration handler 306 sets the audible notification as specified by the user. Continuing to step 428, the configuration handler 306 saves the current configuration parameters as changed by the user. After which, the configuration handler 306 returns to step 402 and reenters the processing loop described above.

Referring again to step 424, if the configuration handler 302 determines that the user did not enter the set audible notification method, the configuration handler 306 proceeds to step 430. In step 430, the configuration handler 306 determines whether the user input is the quit command. If the user entered the quit command, the configuration handler 306 proceeds to step 432. In step 432, the configuration handler 306 processes currently active GUI objects according to new configurations upon exit from the configuration handler 306. Step 432 is discussed in detail below. Step 432 continues to step 434 where the GUI of the configuration handler
306 is terminated. Step 434 then continues to step 436. In step 436, the configuration handler 306 has completed its processing and returns control to step 318 in FIG. 3C.

Referring again to step 430, if the configuration handler 306 determines that the user did not enter the quit command, the configuration handler 306 returns to step 404 and waits for another user input.

6. Control Flow for the Processing of the GUI according to configurations upon exit from the Configuration Handler 306

FIG. 5 is a control flow diagram of step 432 in FIG. 4 and the searching of objects in the graphical user interface 304 according to the processing of the GUI according to configurations upon exit from the configuration handler 306. More specifically, FIG. 5 describes final processing of the configuration handler 306 that searches objects and invokes the perusal list handler 310 to build a corresponding perusal list. Any objects that are found to contain the search criteria are made known to the user. FIG. 5 is processing that resides in the object handler 308. As such, step 432 of FIG. 4, which is embodied by FIG. 5, takes place as processing in the object handler 308. Control begins at step 500 and immediately continues to step 502. In step 502, the object handler 308 invokes the semaphore handler 312 to lock the semaphore regulating access to the shared variables. The semaphore handler 312 is described above and is well known by one of ordinary skill in the art.

Continuing to step 504, the object handler 308 assigns the JUMP_FROM variable a null handle. After which, the object handler 308 continues to step 506 in which it loops through all of the objects in the graphical user interface 304. For each object, the object handler 308 determines whether the object contains the search criteria. Search criteria may include, but is not limited to, the following forms, or combination of forms: text matching, color matching, shade matching, optical character recognition (OCR) of graphic window fields, or graphic matching.
It would be readily apparent to one skilled in the relevant art to use alternative search criteria.

Referring again to step 506, if the object contains the search criteria, the object handler 308 invokes the perusal list handler 310 to add the object to the perusal list. The object is added as an entry to the perusal list. The entry contains the object handle and the attribute settings which were automatically changed (for those attributes configured in step 407 of FIG. 4) so the object can be restored to its normal appearance when it no longer contains the search criteria. The object handler 308 then sets the object attributes to indicate that the object contains the search criteria. The attributes are set according to user preferences in step 407 of FIG. 4. Continuing to step 508, the object handler 308 sets the INDEX variable to point to the first object in the perusal list. If the perusal list is empty, the object handler 308 sets the INDEX to zero.

The object handler 308 continues to step 510 to determine whether the perusal list is empty; that is, to determine if the INDEX equals zero. If the INDEX is not zero, the object handler 308 proceeds to step 514. In step 514, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 516, the object handler 308 provides an audible notification to the user to indicate that at least one object has been found that contains the search criteria. The audible notification is provided in a manner as configured by step 426 of FIG. 4. Continuing to step 518, the object handler 308 determines whether the user has selected preemptive or passive mode. If the user selected preemptive mode, the object handler 308 proceeds to step 520. In step 520, the object handler 308 interrupts the user and displays an object in the perusal list. The object displayed is the one to which the INDEX variable is currently pointing (i.e. first one). Step 520 is described in greater detail below. Continuing to step 522, the object handler 308 returns control back to step 432 in FIG. 4.

Referring again to step 518, if the user did not select preemptive mode, but selected passive mode, the object handler 308 proceeds to step 522. In step 522, the object handler 308 returns control back to step 432 in FIG. 4.
Referring again to step 510, if the perusal list handler 310 determines that the INDEX equals zero (the perusal list is empty), the perusal list handler 310 proceeds to step 512. In step 512, the perusal list handler 310 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 522, processing returns back to step 432 in FIG. 4.

FIG. 6 is a control flow diagram representing the displaying/focusing of an object that contains the search criteria to the user, for example, as referenced in step 520. Control begins at step 602 and immediately continues to step 604. In step 604, the object handler 308 updates the INDEX appropriately; that is, the object handler 308 advances the INDEX variable to point to the next object in the perusal list. If INDEX points to the last object in the perusal list, INDEX is set back to the first object in the list. Continuing to step 606, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore which may have been locked in a process which utilizes FIG. 6. Afterwhich, the object handler 308 continues to step 608.

In step 608, the object handler 308 maximizes the object to be displayed according to the last maximized dimensions of the object. The object handler 308 focuses the object in step 610 before continuing to step 612. In step 612, the object handler 308 returns processing back to the step which referenced FIG. 6, for example, step 520 in FIG. 5. The object manipulated in step 608 and 610 is the object maintained by the perusal list according to the value of INDEX at step 602 (prior to step 604 completion).

7. Control Flow for the Perusal List Handler View function

FIG. 7 is a control flow diagram of the perusal list hot-key and perusal list request as described above. The perusal list hot-key allows a user to traverse the objects in a perusal list in a round-robin fashion, whereas the perusal list request allows a user to view a specific object in the perusal list. The preferred embodiment of the present invention for invoking the viewing of an object in the perusal list comprises a hot-key. A hot-key is a specific key, or combination of
keys, that a user presses to execute the function of viewing an object in the perusal list. It should be understood that the invention is described with reference to a hot-key for convenience purpose only. Other mechanisms for invoking the viewing of an object in the perusal list could alternatively be used, such as a mouse selection, and would be readily apparent to one skilled in the art.

Control begins at step 702 and immediately continues to step 704. In step 704, the object handler 308 invokes the semaphore handler 312 to lock the semaphore. Continuing to step 706, the object handler 308 determines whether the INDEX equals zero; that is, if the perusal list is empty. If the INDEX equals zero then the perusal list is empty and the object handler 308 proceeds to step 716.

In step 716, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 718, the object handler 308 provides the user with an error tone to indicate that there are no objects in the perusal list. The object handler 308 continues to step 716 in which it exits its processing.

Referring again to step 706, if the INDEX does not equal zero, then there are objects in the perusal list which the user may view. Proceeding to step 708, the object handler determines whether the JUMP_FROM variable contains a null handle; that is, if the JUMP_FROM variable has been previously set. If JUMP_FROM contains a null handle, the object handler 308 proceeds to step 710.

In step 710, the object handler 308 saves the handle of the current object that the user is viewing. The object handler 308 continues to step 712.

The JUMP_FROM variable remains loaded with the handle of the object of which the user left when viewing the perusal list until the user hot-keys a return back to the object in the JUMP_FROM variable.

Referring again to step 708, if the JUMP_FROM variable does not contain the null handle, then the user must be in the middle of viewing the objects in the perusal list because the JUMP_FROM variable is already set. Therefore, the object handler 308 proceeds to step 712.

In step 712, the object handler 308 invokes the perusal list handler 310 and gets the next object handle from the perusal list according to the INDEX. Continuing to step 714, the object handler 308 displays the object corresponding
to the object handle that it just received from the perusal list handler 310. Step 714 is identical to step 520 described above and therefore executes processing of FIG. 6. Continuing from step 714, the object handler 308 enters step 716 and exits processing.

8. Control Flow for the Object Handler Original Object Expose function

FIG. 8 is a control flow diagram of the jump-from hot-key as described above. The jump-from hot-key allows a user to return back to the object which the user was previously viewing at the time of initiating the view perusal list hot-key. The preferred embodiment of the present invention for invoking the viewing of the user's original object in the perusal list comprises a hot-key. A hot-key is a specific key, or combination of keys, that a user would press to execute the function of viewing the user's original object in the perusal list. It should be understood that the invention is described with reference to a hot-key for convenience purpose only. Other mechanisms for invoking the viewing of the user's original object in the perusal list could alternatively be used, such as a mouse selection, and it would be readily apparent to one skilled in the art to develop alternative mechanisms.

Control begins at step 802 and immediately continues to step 804. In step 804, the object handler 308 invokes the semaphore handler 312 to lock the semaphore. Continuing to step 806, the object handler 308 determines whether the JUMP_FROM variable equals the null handle; that is, if there is an object to which the object handler 308 can jump. If the JUMP_FROM variable equals the null handle, the object handler 308 proceeds to step 814. In step 814, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 816, the object handler exits its processing.

Referring again to step 806, if the JUMP_FROM variable does not equal the null handle, the object handler 308 continues to step 808. In step 808, the object handler maximizes the object identified in the JUMP_FROM variable according to the last maximized dimensions. Continuing to step 810, the object
handler 308 focuses the object. Continuing to step 812, the object handler 308 sets the JUMP_FROM variable to the null handle before entering step 814. In step 814, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 816, the object handler exits its processing. Note that the JUMP_FROM variable remains persistent (see step 708 of FIG. 7) until the JUMP_FROM hot-key is invoked.

9. Control Flow for the Object Handler Create function

FIG. 9 is a control flow diagram of the object handler 308 creating an object. Step 902 is started for every GUI object which is created (by the user or automatically) within the system. Control begins at step 902 and immediately continues to step 904. In step 904, the object handler 308 determines whether the new object contains the search criteria. If the object does not contain the search criteria, the object handler proceeds to step 918. In step 918, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 922, the object handler 308 exits its processing.

Referring again to step 904, if the object handler 308 determines that the object does contain the search criteria, it proceeds to step 906. In step 906, the object handler 308 invokes the semaphore handler 312 to lock the semaphore. Continuing to step 908, the object handler 308 invokes the perusal list handler 310 to add the object to the perusal list. The object handler 308 also updates the object’s attributes to indicate that it contains the search criteria. The object is added as an entry to the perusal list. The entry contains the object handle and the attribute settings which were automatically changed (for those attributes configured in step 407 of FIG. 4) so the object can be restored to its normal appearance when it no longer contains the search criteria. The attributes are set according to user preferences in step 407 of FIG. 4. Continuing to step 910, the object handler 308 provides an audible notification to the user as another indication that the object contains the search criteria. The audible notification is provided in a manner as
configured by step 426 of FIG. 4. Step 908 did not necessarily affect a visually perceptible object (e.g. overlaid window).

The object handler 308 continues to step 912 in which it determines whether the new object is the only object in the perusal list. If the object is the only one in the perusal list, the object handler proceeds to step 914 and sets the INDEX variable to the value one. Afterwhich, the object handler 308 continues to step 916.

Referring again to step 912, if the object handler 308 determines that there is more than one object in the perusal list, the object handler 308 proceeds to step 916.

In step 916, the object handler 308 determines whether the user had selected preemptive or passive mode. If the user selected preemptive mode, the object handler 308 proceeds to step 920. In step 920, the object handler 308 interrupts the user by displaying the object in the perusal list. The object displayed is the one to which the INDEX variable is pointing. Step 920 is identical to step 520 described above and therefore executes processing of FIG. 6. Continuing to step 922, the object handler 308 exits its processing.

Referring again to step 916, if the user had not selected preemptive mode, but selected passive mode, the object handler 308 proceeds to step 918. In step 918, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 922, the object handler 308 exits its processing.

10. Control Flow for the Object Handler Modify function

FIG. 10 is a control flow diagram of the object handler 308 changing an object. Step 1002 is started for every GUI object which is modified (by the user or automatically) within the system. Control begins at step 1002 and immediately continues to step 1004. In step 1004, the object handler 308 determines whether the changed object contains the search criteria. If the object does not contain the search criteria, the object handler proceeds to step 1006. In step 1006, the object handler 308 invokes the semaphore handler 312 to lock the semaphore. Continuing
to step 1008, the object handler 308 invokes the perusal list handler 310 to
determine whether the perusal list contains the object. If the perusal list does not
contain the object, the object handler 308 proceeds to step 1012 in which the object
handler 308 invokes the semaphore handler 312 to unlock the semaphore. The
object handler 308 then continues to step 1020 and exits its processing.

Referring again to step 1008, if the perusal list handler 310 determines that
the perusal list contains the object, the object handler 308 proceeds to step 1010.
In step 1010, the object handler 308 invokes the perusal list handler 310 to remove
the object from the perusal list. Step 1010 is described in greater detail below.
Continuing to step 1012, the object handler 308 invokes the semaphore handler 312
to unlock the semaphore. The object handler 308 then continues to step 1020 and
exits its processing.

Referring again to step 1004, if the object handler 308 determines that the
object does contain the search criteria, it proceeds to step 1014. In step 1014, the
object handler 308 invokes the semaphore handler 312 to lock the semaphore.
Continuing to step 1016, the object handler 308 invokes the perusal list handler 310
to determine whether the perusal list contains the object. If the perusal list does
contain the object, the object handler 308 proceeds to step 1012 in which the object
handler 308 invocations the semaphore handler 312 to unlock the semaphore. The
object handler 308 then continues to step 1020 and exits its processing.

Referring again to step 1016, if the perusal list handler 310 determines that
the perusal list does not contain the object, the object handler 308 proceeds to step
1018. In step 1018, the object handler 308 invokes the perusal list handler 310 to
add the object to the perusal list. Step 1018 is described in greater detail below.
Continuing to step 1020, the object handler 308 exits its processing.

FIG. 11 is a control flow diagram of step 1010 and illustrates how the
object handler 308 and the perusal list handler 310 remove an object from the
perusal list. Control begins in step 1102 and immediately continues to step 1104.
In step 1104, the object handler 308 invokes the perusal list handler 310 to remove
the object from the perusal list. After which, the object handler 308 restores the
object's attributes to its original values using the information stored in the perusal
list entry. Continuing to step 1106, the object handler 308 determines whether the perusal list is empty. If the perusal list is empty, the object handler 308 proceeds to step 1108 and sets the INDEX variable to zero. Continuing to step 1110, the object handler 308 returns control back to step 1010 in FIG. 10.

Referring again to step 1106, if the object handler 308 determines that the perusal list is not empty after removing the object, it proceeds to step 1112. In step 1112, the object handler 308 sets the INDEX appropriately to point to the perusal list. If the object removed was the last object in the perusal list, then the INDEX variable is set to the first object in the perusal list. Otherwise, INDEX remains unchanged by step 1112. Continuing to step 1110, the object handler 308 returns control back to step 1010 in FIG. 10.

FIG. 12 is a control flow diagram of step 1018 and illustrates how the object handler 308 and the perusal list handler 310 add an object to the perusal list. Control begins in step 1202 and immediately continues to step 1204. In step 1204, the object handler 308 invokes the perusal list handler 310 to add the object to the perusal list. After which, the object handler 308 updates the object's attributes to reflect that the object contains the search criteria. The object is added as an entry to the perusal list. The entry contains the object handle and the attribute settings which were automatically changed (for those attributes configured in step 407 of FIG. 4) so the object can be restored to its normal appearance when it no longer contains the search criteria. The attributes are set according to the user preferences in step 407 of FIG. 4. Continuing to step 1206, the object handler 308 provides the user with an audible notification as another indication that the object contains the search criteria. The audible notification is provided in a manner as configured by step 426 of FIG. 4.

The object handler 308 continues to step 1208 in which it determines whether the new object is the only object in the perusal list. If the object is the only one in the perusal list, the object handler proceeds to step 1210 and sets the INDEX variable to the value one. After which, the object handler 308 continues to step 1212.
Referring again to step 1208, if the object handler 308 determines that there is more than one object in the perusal list, the object handler 308 proceeds to step 1212.

In step 1212, the object handler 308 determines whether the user had selected preemptive or passive mode. If the user had selected preemptive mode, the object handler 308 proceeds to step 1218. In step 1218, the object handler 308 interrupts the user and displays the object just added to the perusal list. The object displayed is the one to which the INDEX variable is pointing. Step 1218 is identical to step 520 described above and therefore executes processing of FIG. 6. Continuing to step 1216, the object handler 308 returns control to step 1018 in FIG. 10.

Referring again to step 1212, if the user did not select preemptive mode, but selected passive mode, the object handler 308 proceeds to step 1214. In step 1214, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. Continuing to step 1216, the object handler 308 returns control to step 1018 in FIG. 10.

11. Control Flow for the Object Handler Remove function

FIG. 13 is a control flow diagram of the object handler 308 deleting an object. Step 1302 is started for every GUI object which is deleted or removed (by the user or automatically) within the system. Control begins at step 1302 and immediately continues to step 1304. In step 1304, the object handler 308 invokes the semaphore handler 312 to lock the semaphore. Continuing to step 1306, the object handler 308 invokes the perusal list handler 310 to determine whether the perusal list contains the object to be deleted. If the perusal list does not contain the object, the object handler 308 proceeds to step 1316. In step 1316, the object handler deletes the object from the graphical user interface 304 and removes all references to the object from the presentation manager and operating system. Continuing to step 1318, the object handler 308 invokes the semaphore handler 312
to unlock the semaphore. After which, the object handler 308 enters step 1320 and exits its processing.

Referring again to step 1306, if the perusal list handler 310 determines that the perusal list contains the object to be deleted, the object handler 308 proceeds to step 1308. In step 1308, the object handler 308 invokes the perusal list handler 310 to remove the object from the perusal list. Continuing to step 1310, the object handler determines whether the perusal list is empty. If the perusal list is empty, the object handler proceeds to step 1312 and sets the INDEX variable to zero. Continuing to step 1316, the object handler deletes the object from the graphical user interface 304 and removes all references to the object from the presentation manager and operating system. After which, in step 1318, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. The object handler 308 then enters step 1320 and exits its processing.

Referring again to step 1310, if the object handler 308 determines that the perusal list is not empty, the object handler proceeds to step 1314 and sets the INDEX variable appropriately. If the object removed was the last object in the perusal list, the object handler 308 sets the INDEX variable to the first object in the perusal list. Otherwise, the INDEX variable remains unchanged by step 1314. Continuing to step 1316, the object handler deletes the object from the graphical user interface 304 and removes all references to the object from the presentation manager and operating system. After which, in step 1318, the object handler 308 invokes the semaphore handler 312 to unlock the semaphore. The object handler 308 then enters step 1320 and exits its processing.

**Conclusion**

While various embodiments of the present invention have been described above, it should be understood that they have been presented by the way of example only, and not limitation. It will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined in the appended claims. Thus,
the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined in accordance with the following claims and their equivalents.
What Is Claimed Is:

1. A system for proactively searching a plurality of objects in a graphical user interface, said system comprising:
   a configuration means for configuring search criteria;
   a searching means for automatically searching the plurality of objects to determine whether one or more objects contain said search criteria, said objects are automatically searched upon creation, deletion, or modification, and wherein said objects that contain said search criteria are stored in a perusal list; and
   displaying means for displaying said objects stored in said perusal list.

2. The system according to claim 1, wherein one of said objects is an Internet address.

3. The system according to claim 1, wherein said perusal list comprises one or more entries, each entry comprising a graphical user interface handle of an object in said graphical user interface and one or more attributes associated with said object in said graphical user interface.

4. The system according to claim 1, wherein said searching means automatically searches forthcoming objects of said graphical user interface to determine whether said forthcoming objects contain said search criteria.

5. The system according to claim 1, further comprising a preemptive means for preempting an executing process in said graphical user interface and displaying an object that contains said search criteria when said object contains said search criteria.
6. The system according to claim 1, further comprising a passive means for allowing normal processing in said graphical user interface to continue without interruption until a user requests to view an object in said perusal list.

7. The system according to claim 1, wherein said displaying means comprises a perusal list hot-key means for traversing said perusal list in a round-robin fashion and displaying said objects contained in said perusal list.

8. The system according to claim 1, wherein said displaying means comprises a perusal list request means for displaying a specific object in said perusal list.

9. The system according to claim 7, wherein said displaying means further comprises a jump-from means for displaying an object that a user left prior to invoking said perusal list hot-key means.

10. The system according to claim 1, wherein said searching means comprises an object attribute means for automatically changing one or more attributes of an object when said searching means determines that said object contains said search criteria.

11. The system according to claim 10, wherein said searching means further comprises a restore object means for restoring said one or more attributes of said object to an original setting when said searching means determines that said object no longer contains said search criteria, wherein said one or more attributes of said object were automatically changed when said searching means determined that said object contained said search criteria.

12. The system according to claim 1, wherein said objects in said graphical user interface represent a plurality of executing processes in a multi-tasking
environment, said multi-tasking environment using semaphores to regulate access to resources of said system.

13. The system according to claim 1, wherein said searching means comprises a notification means for automatically notifying a user when said searching means determines that an object contains said search criteria.

14. The system according to claim 13, wherein said notification means comprises notifying said user with a configurable audible tone.

15. A method of proactively searching a plurality of objects in a graphical user interface, said method comprising:
   (1) configuring search criteria;
   (2) searching the plurality of objects to determine whether one or more objects contain said search criteria, said searching occurs automatically upon the creation, deletion, or modification of an object, wherein upon a determination that said object contains said search criteria, said object is stored in a perusal list; and
   (3) displaying said objects stored in said perusal list.

16. The method of claim 15, wherein step (3) comprises:
   (a) determining whether a user selected a preemptive mode;
   (b) preempting an executing process and displaying an object that contains said search criteria if it is determined in step (3.a) that said user selected said preemptive mode, and
   (c) displaying an object in said perusal list only upon a user request if is determined in step (3.a) that said user did not select said preemptive mode.

17. The method of claim 15, wherein step (3) comprises:
   (a) determining whether a user selected a passive mode;
32

(b) allowing normal processing to continue without interruption;

(c) determining whether a user requests to view an object in said perusal list; and

(d) displaying said object in said perusal list if it is determined in step (3.c) that said user requests to view said object.

18. The method of claim 15, wherein step (3) comprises:

(a) receiving a perusal list hot-key, and

(b) displaying an object in said perusal list upon the receipt of said perusal list hot-key in step (3.a), wherein upon successive receipts of said perusal list hot-key said objects of said perusal list are viewed in a round-robin fashion.

19. The method of claim 15, wherein step (3) comprises:

(a) receiving a view perusal list request; and

(b) displaying a specific object in said perusal list upon the receipt of said view perusal list request in step (3.a).

20. The method of claim 18, wherein step (3) further comprises:

(c) receiving a jump-from hot-key; and

(d) displaying an object that a user left prior to invoking said perusal list hot-key upon the receipt of said jump-from hot-key in step (3.c).

21. The method of claim 15, wherein step (3) comprises:

(a) changing one or more attributes of an object when it is determined in step (2) that said object contains said search criteria.
22. The method of claim 21, wherein step (3.a) comprises drawing said object’s border thicker when it is determined in step (2) that said object contains said search criteria.

23. The method of claim 21, wherein step (3.a) comprises highlighting said search criteria in said object when it is determined in step (2) that said object contains said search criteria.

24. The method of claim 21, wherein step (3) further comprises:
   (b) restoring said one or more attributes of said object to an original setting when step (3) determines that said object no longer contains said search criteria, wherein said one or more attributes of said object were changed in step (3.a).

25. The method of claim 15, further comprising:
   (4) notifying a user with an audible tone when it is determined in step (2) that said object contains said search criteria.

26. The method of claim 15, wherein step (2) comprises:
   (a) searching automatically forthcoming objects of said graphical user interface to determine whether said forthcoming objects contain said search criteria, and wherein said forthcoming objects that contain said search criteria are stored in said perusal list.

27. A computer program product comprising:
   a computer usable medium having computer readable program code means embodied in said medium for proactively searching a plurality of objects in a graphical user interface, said computer program product including:
   first computer readable program code means for configuring search criteria;
second computer readable program code means for automatically searching the plurality of objects to determine whether one or more objects contain said search criteria, said objects are automatically searched upon creation, deletion, or modification, and wherein said objects that contain said search criteria are stored in a perusal list, and

third computer readable program code means for displaying said objects stored in said perusal list.

28. The computer program product according to claim 27, wherein said perusal list comprises one or more entries, each entry comprising a graphical user interface handle of an object in said graphical user interface and one or more attributes associated with said object in said graphical user interface.

29. The computer program product according to claim 27, wherein second computer readable program code means automatically searches forthcoming objects of said graphical user interface to determine whether said forthcoming objects contain said search criteria.

30. The computer program product according to claim 27, wherein said third computer readable program code means comprises a perusal list hot-key means for traversing said perusal list in a round-robin fashion and displaying said objects contained in said perusal list.

31. The computer program product according to claim 27, wherein said third computer readable program code means comprises a perusal list request means for displaying a specific object in said perusal list.

32. The computer program product according to claim 30, wherein said third computer readable program code means further comprises a jump-from means for displaying an object that a user left prior to invoking said perusal list hot-key means.
33. The computer program product according to claim 27, wherein said second computer readable program code means comprises an object attribute means for automatically changing one or more attributes of an object when said second computer readable program code means determines that said object contains said search criteria.

34. The computer program product according to claim 33, wherein said second computer readable program code means further comprises a restore object means for restoring said one or more attributes of said object to an original setting when said second computer readable program code means determines that said object no longer contains said search criteria, wherein said one or more attributes of said object were automatically changed when said second computer readable program code means determined that said object contained said search criteria.

35. The computer program product according to claim 27, wherein said objects in said graphical user interface represent a plurality of executing processes in a multi-tasking environment, said multi-tasking environment using semaphores to regulate access to resources of said system.

36. The computer program product according to claim 27, wherein said second computer readable program code means comprises a notification means for automatically notifying a user when said second computer readable program code means determines that an object contains said search criteria.

37. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for proactively searching a plurality of objects in a graphical user interface, said method steps comprising:

(1) configuring search criteria;
(2) searching the plurality of objects to determine whether one
or more objects contain said search criteria, said searching occurs
automatically upon the creation, deletion, or modification of an object,
wherein upon a determination that said object contains said search criteria,
said object is stored in a perusal list; and

(3) displaying said objects stored in said perusal list.

38. The program storage device of claim 37, wherein step (3) comprises:
   (a) receiving a perusal list hot-key; and
   (b) displaying an object in said perusal list upon the receipt of
   said perusal list hot-key in step (3.a), wherein upon successive receipts of
   said perusal list hot-key said objects of said perusal list are viewed in a
   round-robin fashion.

39. The program storage device of claim 37, wherein step (3) comprises:
   (a) receiving a view perusal list request; and
   (b) displaying a specific object in said perusal list upon the
   receipt of said view perusal list request in step (3.a).

40. The program storage device of claim 38, wherein step (3) further
comprises:
   (c) receiving a jump-from hot-key; and
   (d) displaying an object that a user left prior to invoking said
   perusal list hot-key upon the receipt of said jump-from hot-key in step
   (3.c).

41. The program storage device of claim 37, wherein step (3) comprises:
   (a) changing one or more attributes of an object when it is
determined in step (2) that said object contains said search criteria.
42. The program storage device of claim 41, wherein step (3.a) comprises
drawing said object’s border thicker when it is determined in step (2) that
said object contains said search criteria.

43. The program storage device of claim 41, wherein step (3.a) comprises
highlighting said search criteria in said object when it is determined in step
(2) that said object contains said search criteria.

44. The program storage device of claim 41, wherein step (3) further
comprises:
   (b) restoring said one or more attributes of said object to an
original setting when step (3) determines that said object no longer contains
said search criteria, wherein said one or more attributes of said object were
changed in step (3.a).

45. The program storage device of claim 37, further comprising:
   (4) notifying a user with an audible tone when it is determined
in step (2) that said object contains said search criteria.

46. The program storage device of claim 37, wherein step (2) comprises:
   (a) searching automatically forthcoming objects of said
graphical user interface to determine whether said forthcoming objects
contain said search criteria, and wherein said forthcoming objects that
contain said search criteria are stored in said perusal list.
Start

Configure Search Criteria

Stop

FIG. 3C
FIG. 3D
Start
400
Display Current Configuration Criteria
402
Wait for User Input
404
Set Attributes?
406
Y
Set Object Attributes
407
N
Add Search Criteria?
408
Y
Add Search Criteria
409
N
Delete Search Criteria?
410
Y
Delete Search Criteria
412
N
Change Search Criteria?
414
Y
Change Search Criteria
415
N
Set Mode?
416
Y
Set Search Mode
418
N
Set Audible Notification?
420
Y
Set Audible Notification
422
N
Process GUI According to New Configurations
432
Terminate GUI
434
Return
436
QUIT?
Start 500

Lock Semaphore 502

JUMP_FROM = Null Handle 504

Loop Through Objects and Rebuild Perusal List; Set Attributes of Objects in Perusal List 506

Set INDEX 508

INDEX = 0? 510

Unlock Semaphore 514

Unlock Semaphore 512

Provide Audible Notification to User 516

Search Mode = Preempt? 518

Display Object in Perusal List 520

Return 522

FIG. 5

SUBSTITUTE SHEET (RULE 26)
Start

Lock Semaphore

INDEX = 0?

Unlock Semaphore

Provide User With Error Tone

JUMP_FROM = Null Handle?

Save Handle of Object Currently Focused

Get Object Handle From Perusal List According to INDEX

Display Object in Perusal List

Stop

FIG.7

SUBSTITUTE SHEET (RULE 26)
Start

Object contain configured criteria?

Y

Lock Semaphore
Add object to perusal list and update object attributes appropriately
Provide audible notification to user

N

Is this the only object in list?

Y
Set INDEX to 1

N

Search mode = preempt?

Y
Unlock semaphore

N

Display object in perusal list

Stop

FIG. 9
SUBSTITUTE SHEET (RULE 26)
Start

Remove object from perusal list and restore object's attributes

Perusal list empty?

Set INDEX = 0

Set INDEX appropriately

Return

FIG.11
Start

Add object to perusal list and update object's attributes

Provide audible notification to user

Is this the only object in the perusal list?

Set INDEX = 1

Search mode = preempt?

Unlock semaphore

Display object in perusal list

Return

FIG. 12

SUBSTITUTE SHEET (RULE 26)
Start
- Lock semaphore

Object's handle in perusal list?
- Y: Remove object from perusal list
  - Perusal list empty?
    - Y: Set INDEX to 0
    - N: Set INDEX appropriately
  - Destroy object from system
- Unlock semaphore
- Stop

N

FIG. 13
SUBSTITUTE SHEET (RULE 26)
# INTERNATIONAL SEARCH REPORT

**International application No.**
PCT/US97/07629

## A. CLASSIFICATION OF SUBJECT MATTER

<table>
<thead>
<tr>
<th>IPC(6)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G06F 17/30, 3/14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>US CL</td>
<td>395/335, 968</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)


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

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

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US, A, 5,204,947 (BERNSTEIN et al.) 20 April 1993.</td>
<td>1-46</td>
</tr>
</tbody>
</table>

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

Date of the actual completion of the international search 18 JUNE 1997

Date of mailing of the international search report 9 JUL 1997

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