



US 20060230356A1

(19) **United States**

(12) **Patent Application Publication**

Sauve et al.

(10) **Pub. No.: US 2006/0230356 A1**

(43) **Pub. Date: Oct. 12, 2006**

(54) **SYSTEM AND METHOD FOR SELECTING A TAB WITHIN A TABBLED BROWSER**

Publication Classification

(51) **Int. Cl.**
G06F 3/00 (2006.01)

(52) **U.S. Cl.** **715/777**

(75) Inventors: **Aaron J. Sauve**, Seattle, WA (US);
Cornelis K. Van Dok, Bellevue, WA (US);
Donald J. Lindsay, Mountain View, CA (US);
Tony E. Schreiner, Redmond, WA (US)

Correspondence Address:

LEE & HAYES PLLC

421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

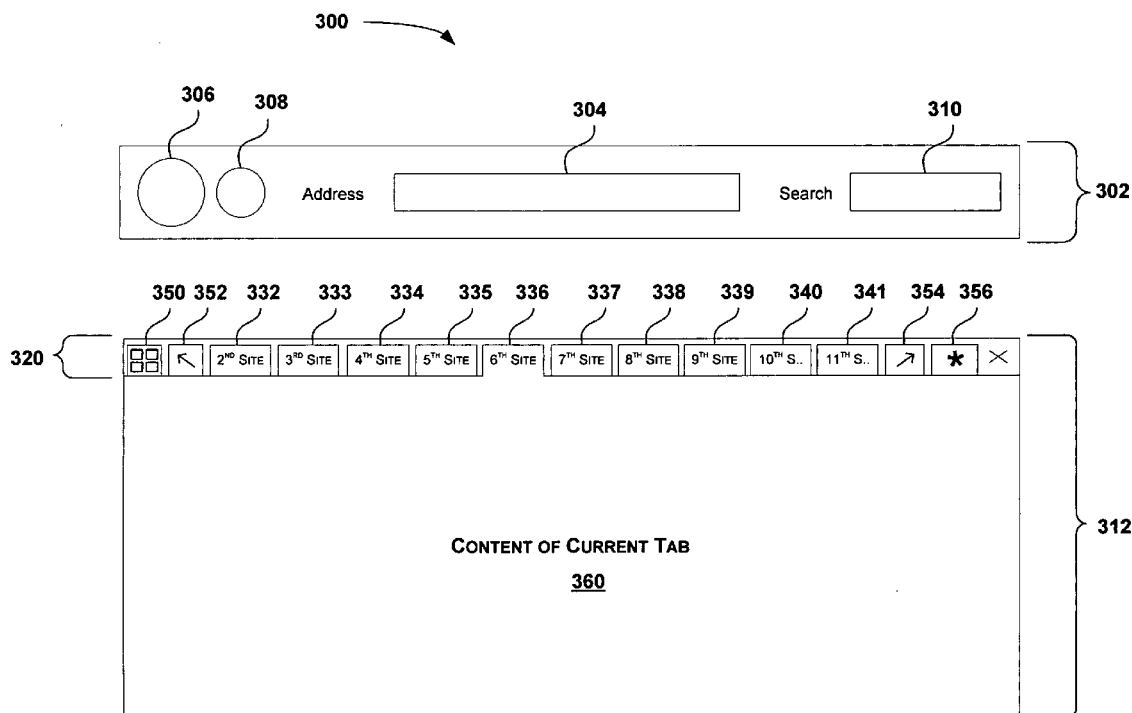
(73) Assignee: **Microsoft Corporation**, Redmond, WA

(21) Appl. No.: **11/101,735**

(22) Filed: **Apr. 7, 2005**

(57) **ABSTRACT**

The techniques and mechanisms described herein are directed to a method for managing and selecting one out of large set of open tabs in a tabbed browser. Browsing software is configured to provide a quick pick user-interface that visually displays the content of the several tabs, such as a by displaying a thumbnail for each tab. The thumbnails allow a user to easily distinguish between each opened tab. The browsing software may also be configured to recognize a user selection from within the quick pick user-interface and in response, switch back to a tabbed window view and display the selected tab in that view. The browsing software may adjust the tab row so that the selected tab is positioned in a favorable position within the tab row. In addition, within the quick pick user-interface, the user may reposition the thumbnails and/or close the thumbnails, thus affecting the associated tab in the tab row.



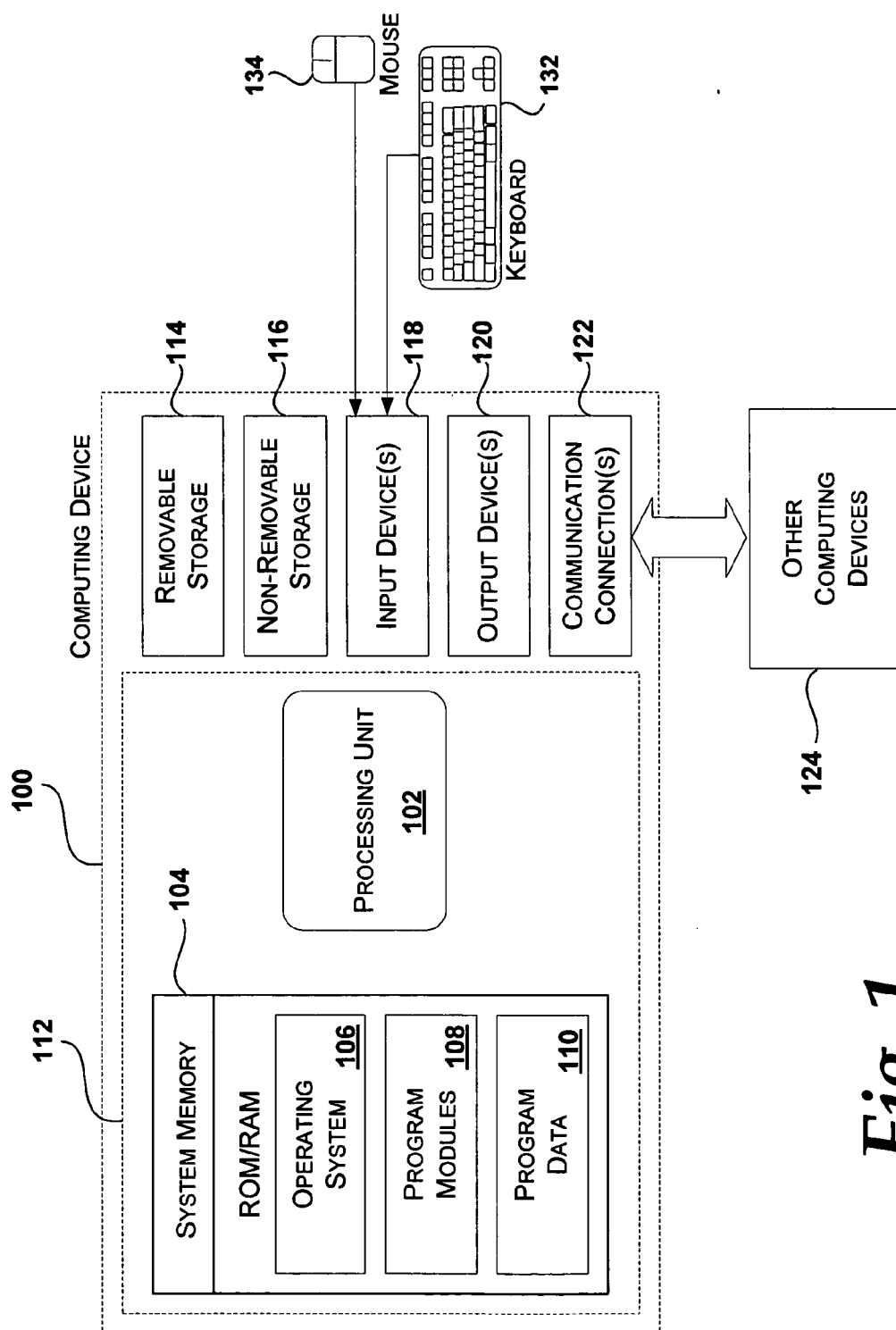
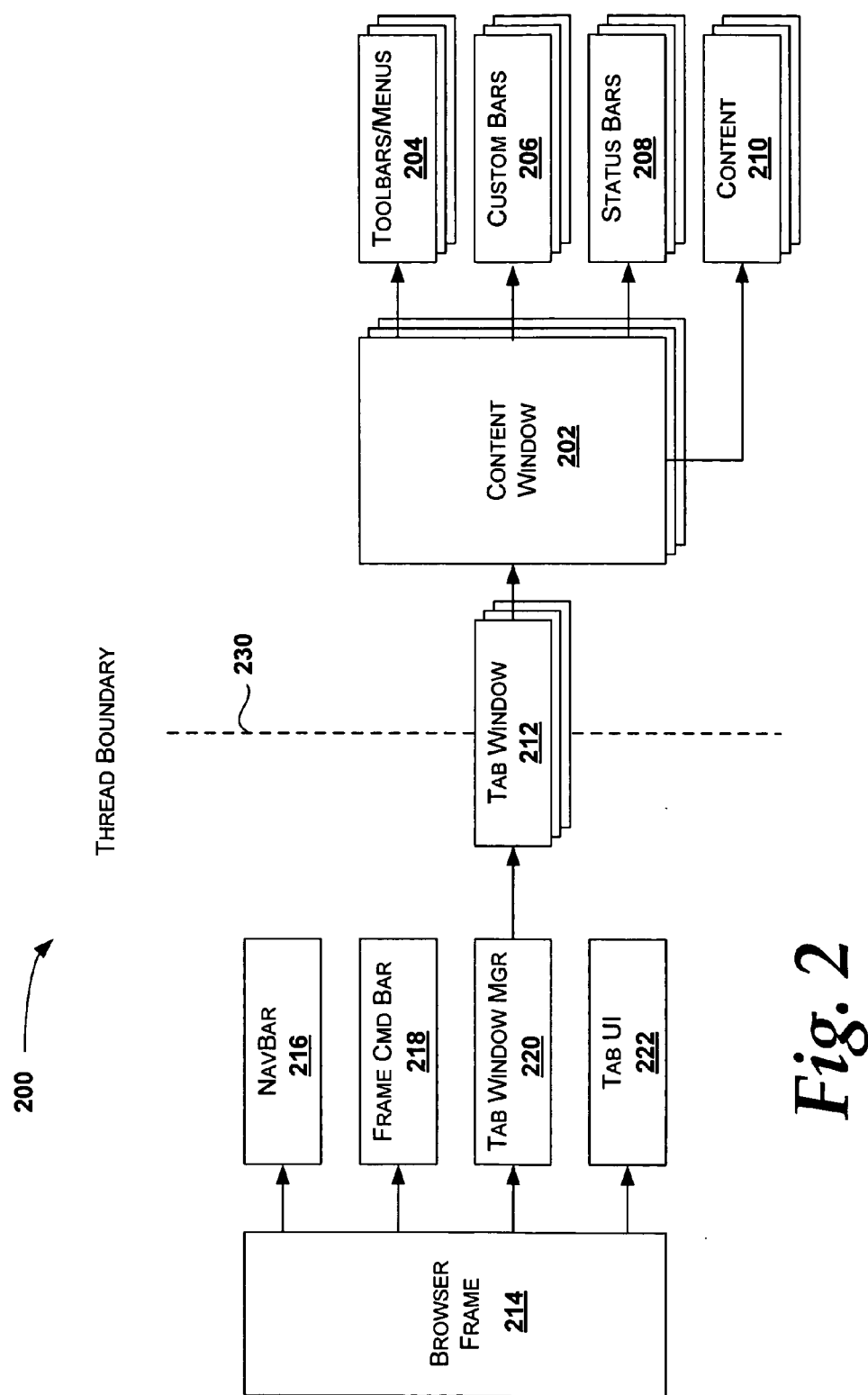


Fig. 1



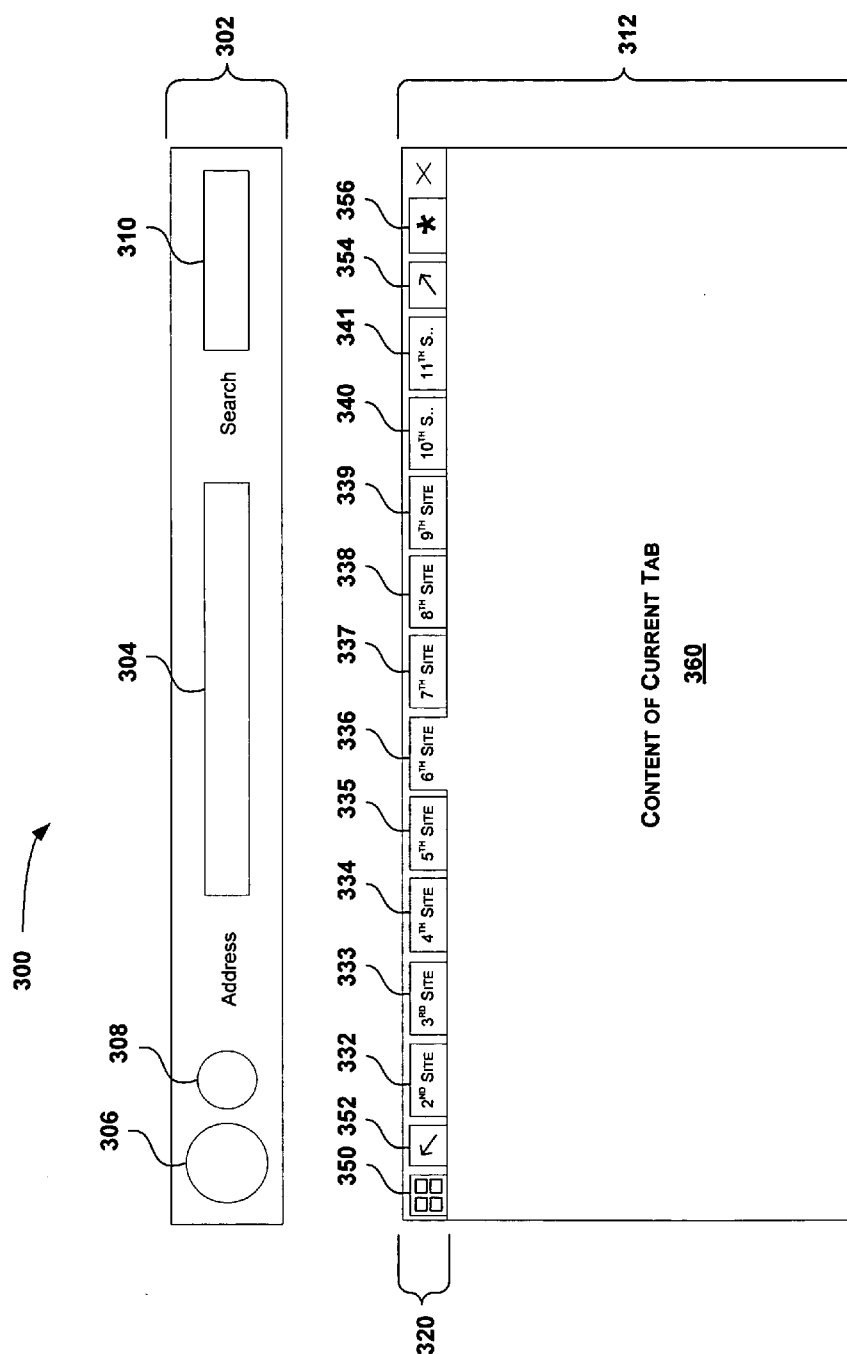


Fig. 3

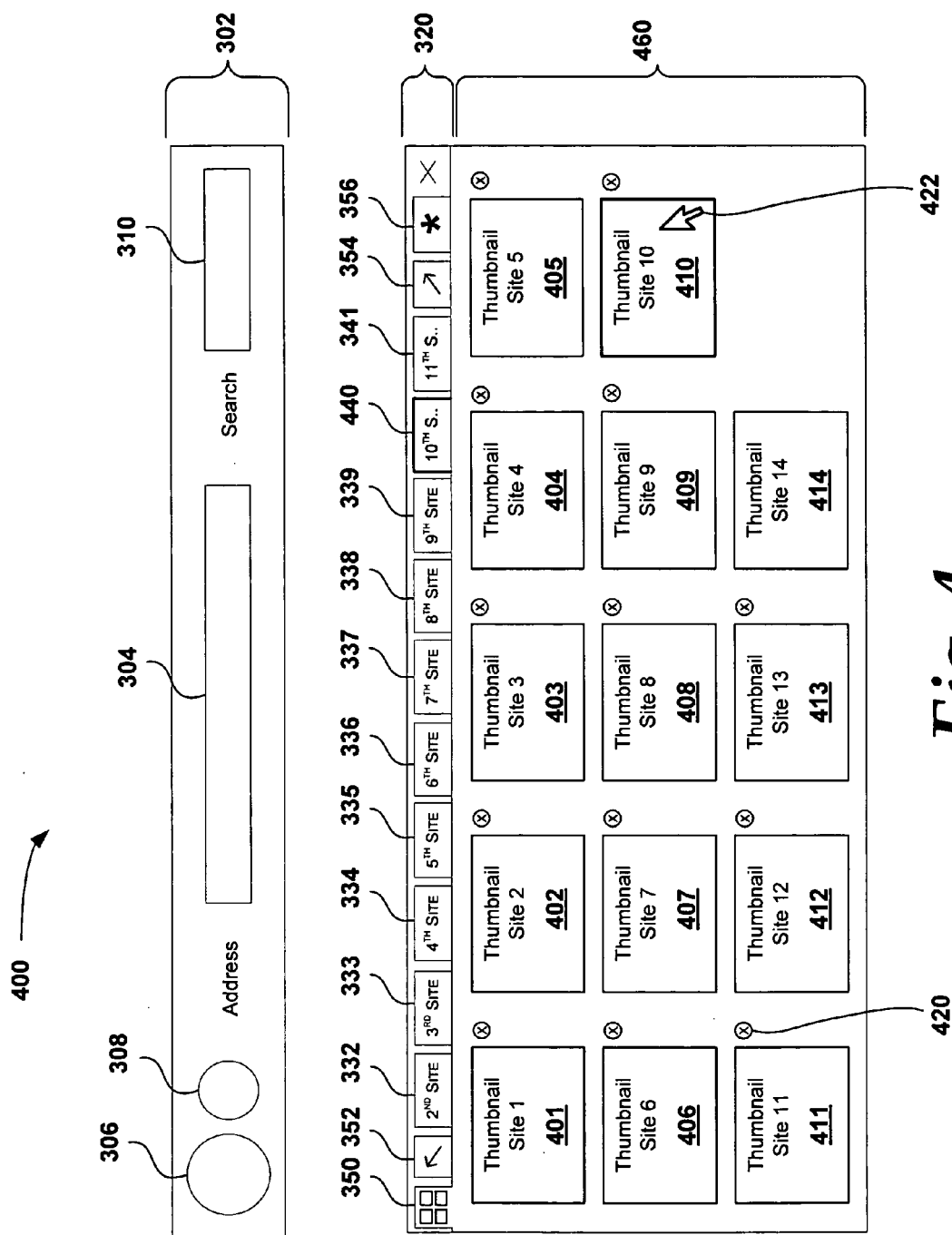


Fig. 4

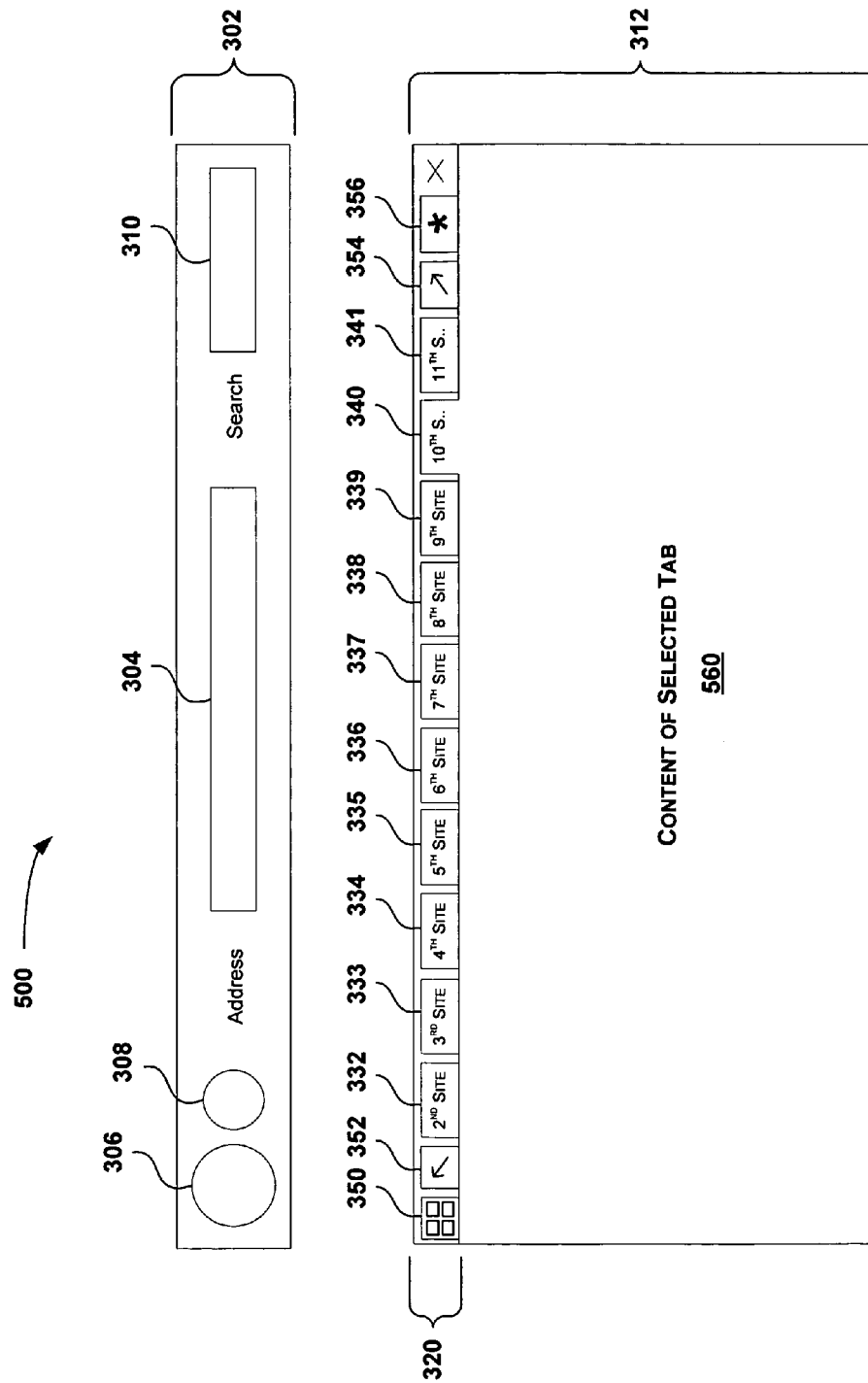


Fig. 5

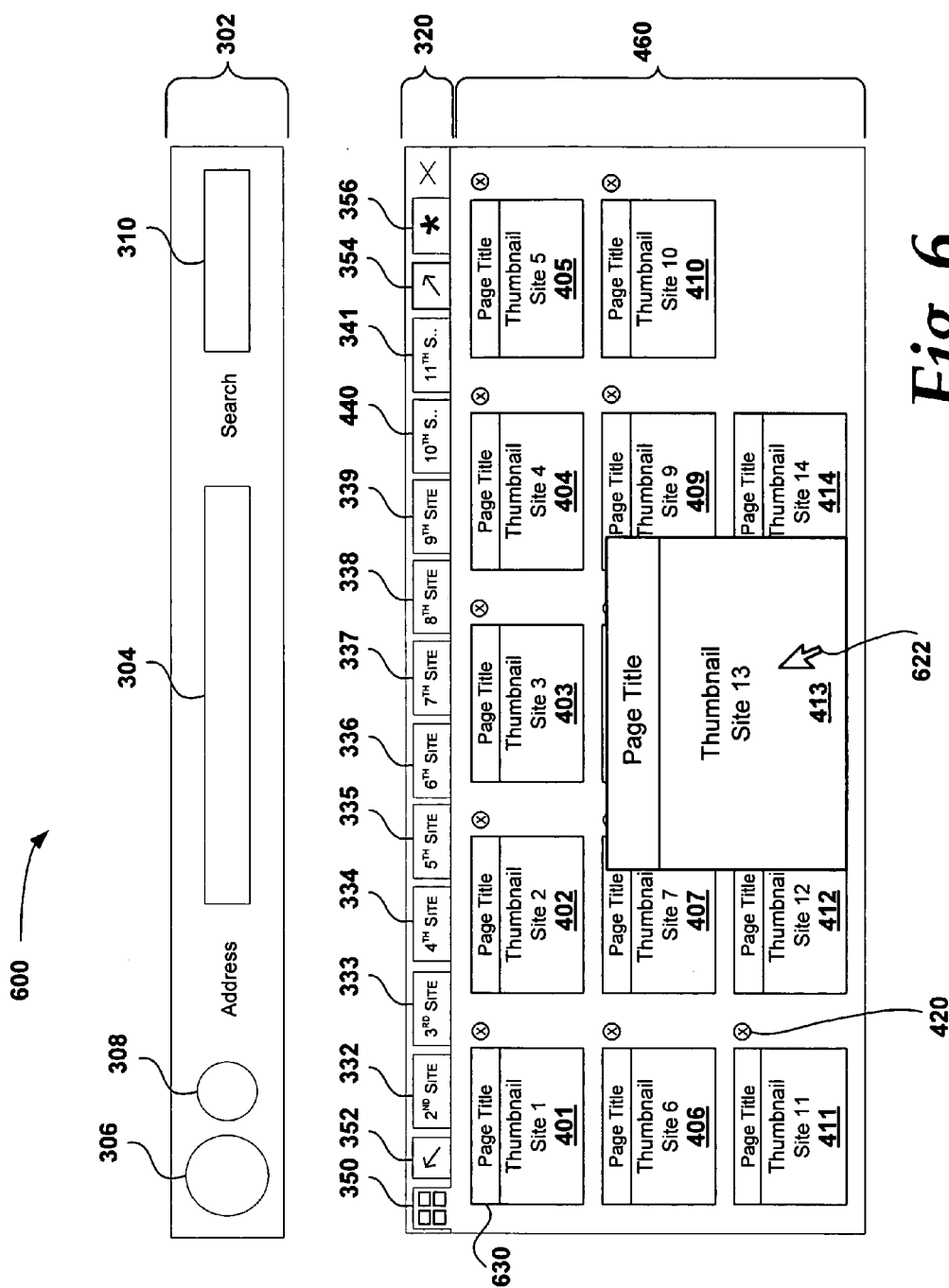


Fig. 6

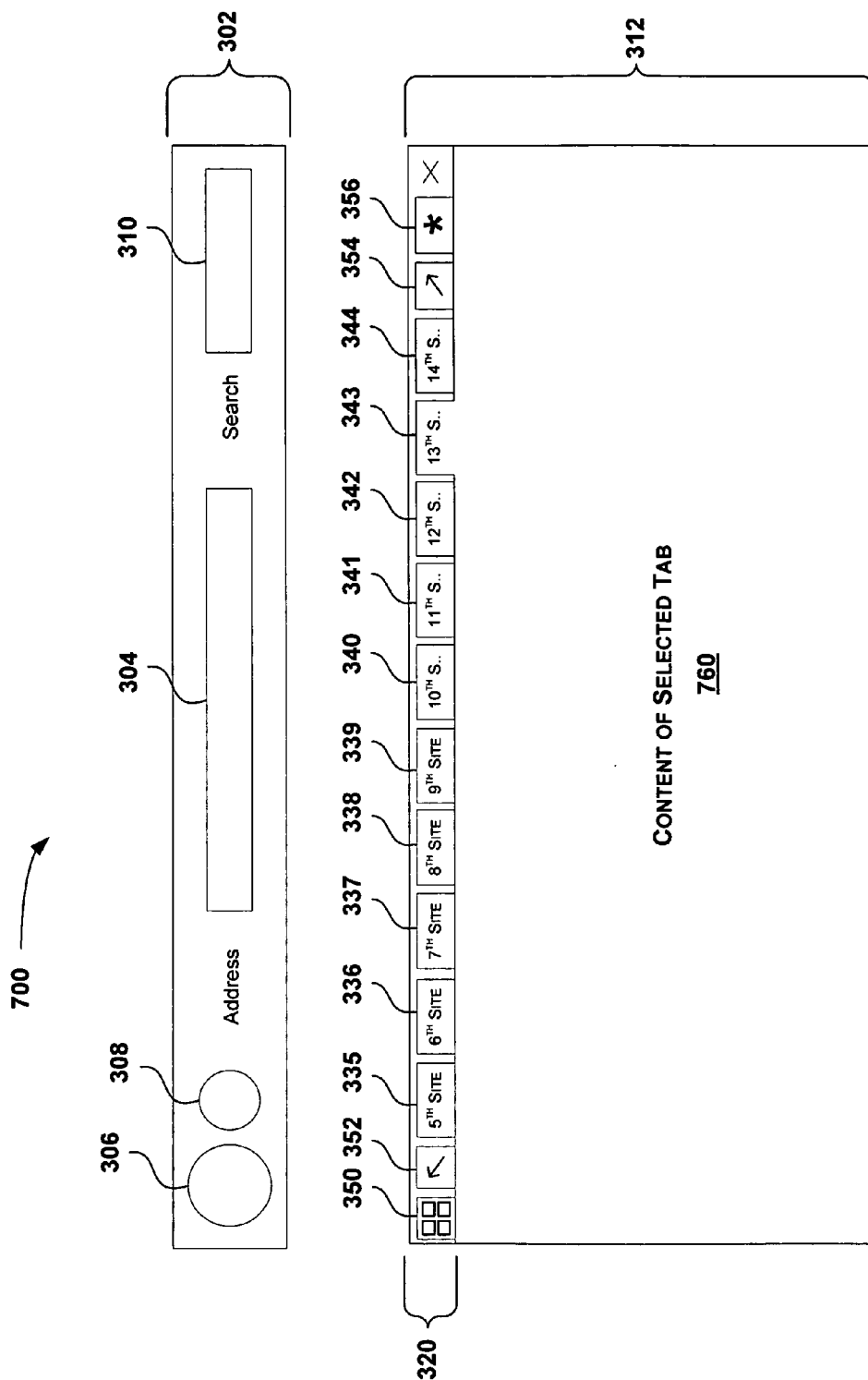


Fig. 7

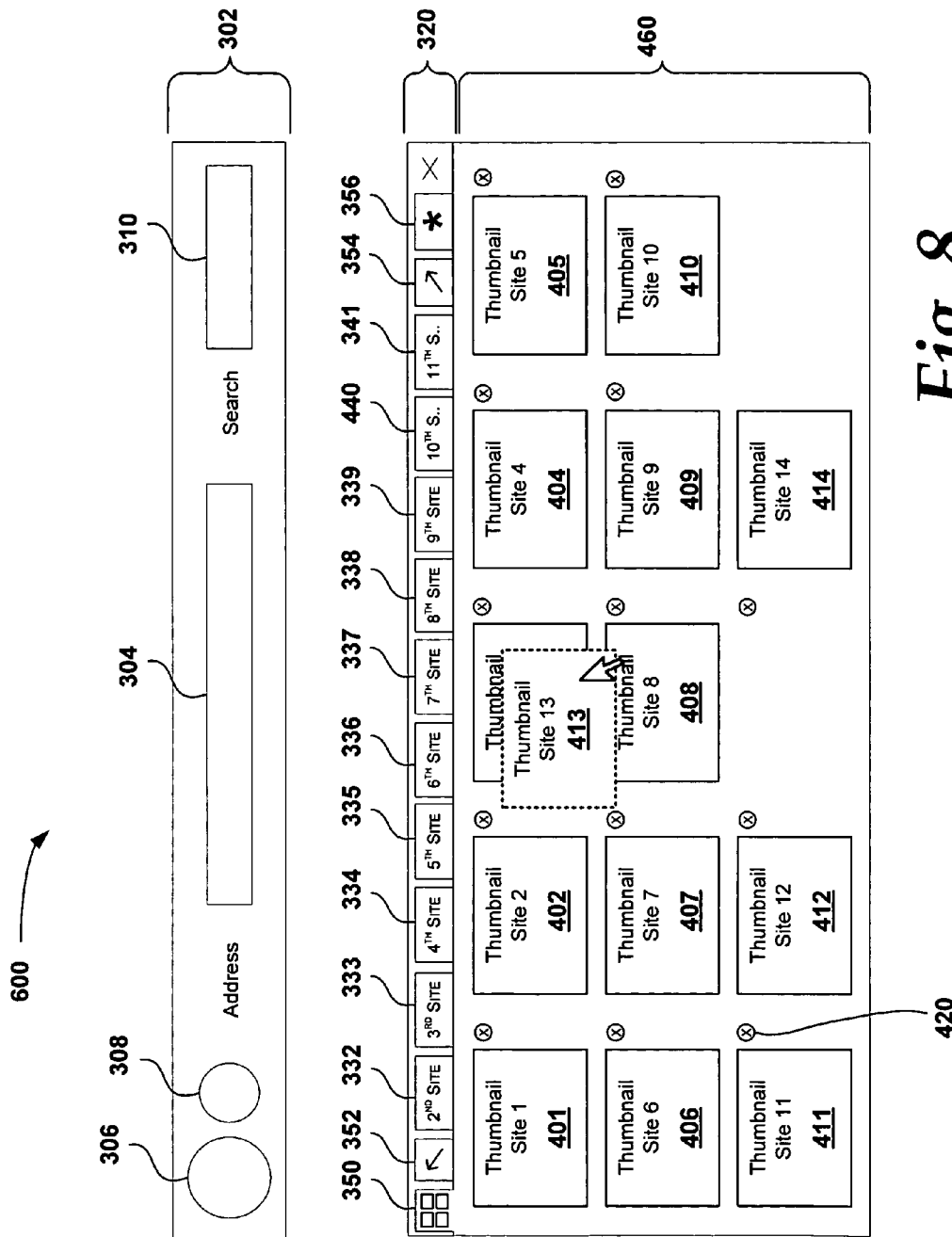


Fig. 8

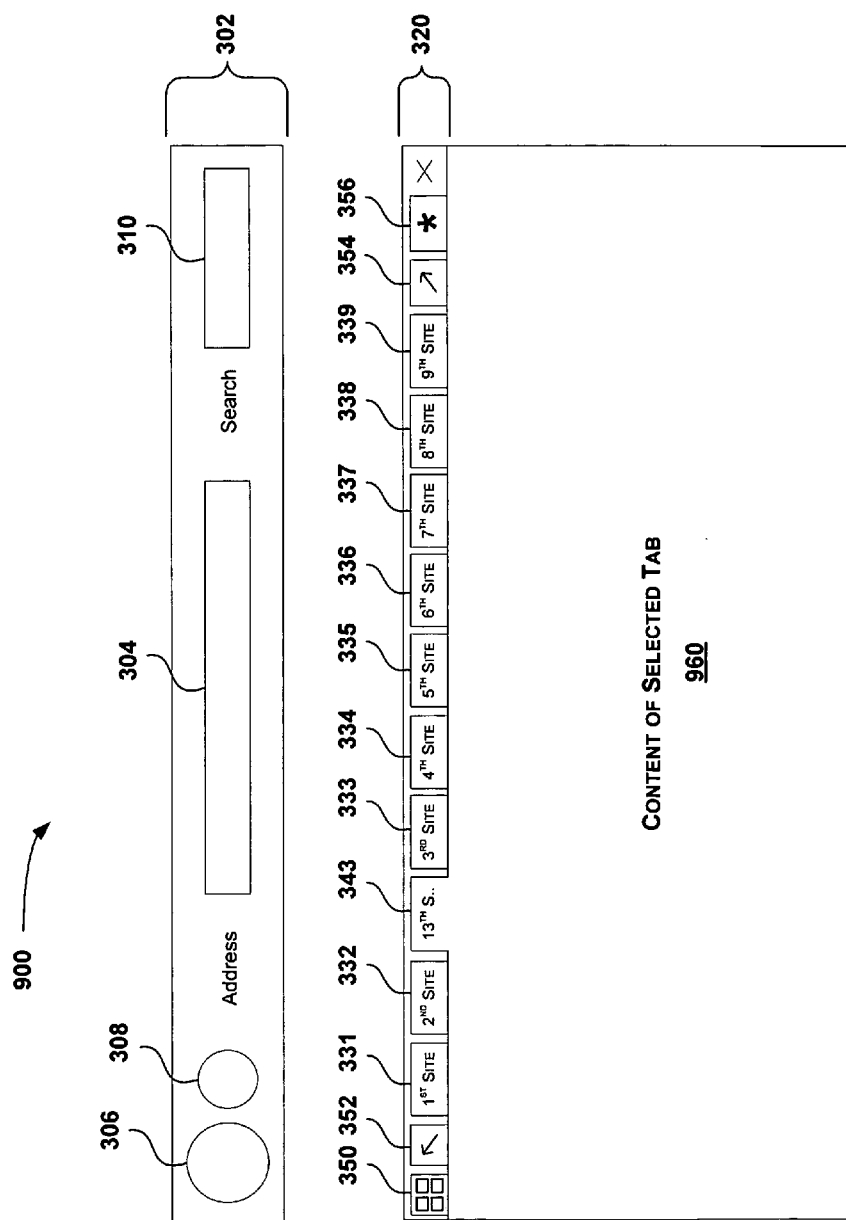


Fig. 9

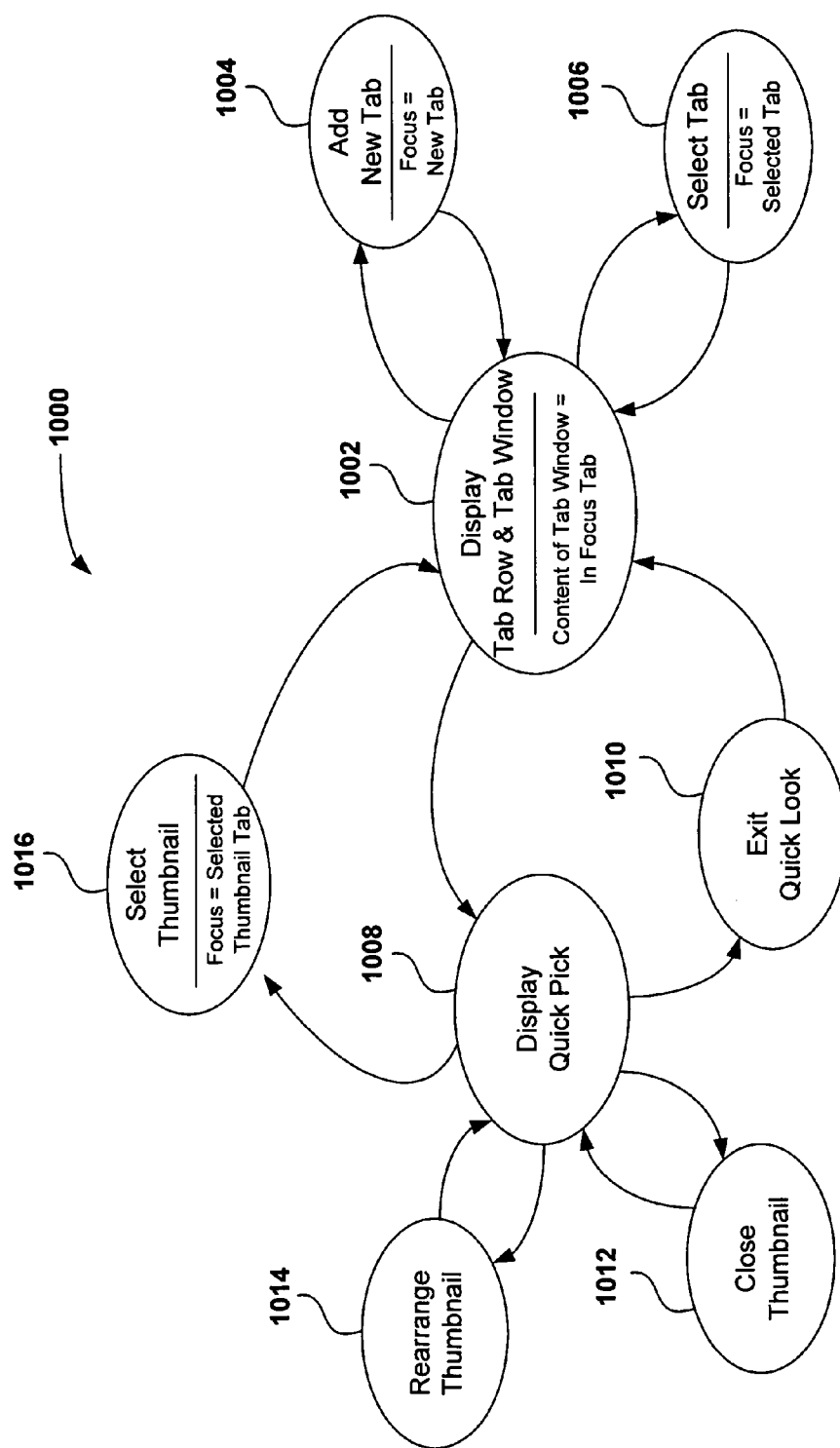


Fig. 10

SYSTEM AND METHOD FOR SELECTING A TAB WITHIN A TABBED BROWSER

TECHNICAL FIELD

[0001] This document generally relates to browsing software, and more particularly, to tabbed-browsing software.

BACKGROUND

[0002] Browsing the World Wide Web (the web), the graphical portion of the Internet, has become quite popular in recent years. A software program, commonly referred to as a web browser, or just browser, provides an easy-to-use point-and-click interface for accessing various content on the web. Upon entering a web address or URL of a particular website, the browser requests web pages from a web server hosting that website. The browser then interprets the web pages and displays the content on a display. The web pages include hypertext and hyperlinks that, when selected, cause the browser to request additional content associated with them. By selecting the hypertext and hyperlinks, a user may conveniently navigate through pages of information—commonly known as browsing or surfing the Internet.

[0003] Each time one of the hypertext or hyperlinks is selected, the new content is downloaded into the current window. Depending on the network bandwidth, this may cause a brief to extensive delay. For convenience, additional windows may be opened to view multiple web pages. However, after opening several web pages, each in its own window, the taskbar may become quite cluttered. This makes it difficult to re-locate a particular web page. Tabbed browsers have been introduced to help manage the viewing of multiple web pages.

[0004] Tabbed browsers load web pages in “tabs” within the same browser window. Therefore, only one item appears on the taskbar, even though multiple web pages are loaded. New tabs may be opened via a hotkey, a context menu item on a link or the like. The new web page downloads in the background into the new tab. This allows a user to continue viewing the current web page without automatically switching to the other web page. At any time, the user may click on the associated tab and view that web page. Tabbed browsing makes it easier and more convenient to view multiple web pages. However, when multiple tabs are open, users may experience difficulty switching between them.

[0005] The present mechanism further enhances the tabbed-browsing experience, especially with selecting one out of a large set of open tabs.

SUMMARY

[0006] The techniques and mechanisms described herein are directed to a method for managing and selecting one out of several open tabs in a tabbed browser. Briefly stated, browsing software is configured to provide a quick pick user-interface that visually displays the several tabs. The quick pick user-interface displays a rich set of information for each tab, such as a thumbnail, meta-data describing each tab (e.g., title) and/or other adornments. The thumbnails allow a user to easily distinguish between each opened tab. The browsing software may also be configured to recognize a user selection from within the quick pick user-interface and in response, switch back to the tabbed window view and

display the selected tab in that view. The browsing software may adjust the tab row so that the selected tab is positioned in a favorable position within the tab row. In addition, within the quick pick user-interface, the user may reposition the thumbnails and/or close the thumbnails, thus affecting the associated tab in the tab row.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Non-limiting and non-exhaustive embodiments are described with reference to the following figures, wherein like reference numerals refer to like parts through-out the various views unless otherwise specified.

[0008] FIG. 1 is an illustrative computer environment that may be used to implement the techniques and mechanisms described herein.

[0009] FIG. 2 is a block diagram illustrating one embodiment of an architecture for implementing tabbed browsing that provides a quick pick tab user-interface as described herein.

[0010] FIG. 3 is a display illustrating one embodiment of a user-interface that supports the quick pick mechanism in a tabbed browser.

[0011] FIG. 4 is a display illustrating one embodiment for the quick pick user-interface and illustrating a user's selection within the interface.

[0012] FIG. 5 is a display illustrating the tabbed window after the user's action within the quick pick user-interface shown in FIG. 4.

[0013] FIG. 6 is a display illustrating the quick pick user-interface and illustrating another user's selection within the interface.

[0014] FIG. 7 is a display illustrating the tabbed window after the user's action within the quick pick user-interface shown in FIG. 6.

[0015] FIG. 8 is a display illustrating the quick pick user-interface and illustrating a user re-positioning a thumbnail within the interface.

[0016] FIG. 9 is a display illustrating the tabbed window after the user's action within the quick pick user-interface shown in FIG. 8.

[0017] FIG. 10 is an operational state diagram generally illustrating the processing for managing open tabs and selecting one out of a large set of open tabs within a tabbed browser using the present quick pick feature.

DETAILED DESCRIPTION

[0018] Briefly, the techniques and mechanisms described herein are directed to managing and selecting one out of a large set of open tabs within a tabbed window displayed by a browser. A quick pick user-interface is provided that visually displays a rich set of information, such as thumbnails, meta-data describing each tab (e.g., title) and the like. The thumbnails may be selected and/or moved within the quick pick user-interface. Upon switching back to the tabbed window view, the tab row and the contents of the tabbed window are modified based on the interactions that occurred in the quick pick user interface. The following discussion describes the quick pick mechanism within a tabbed browser

that supports tabbed-heterogeneous windows. However, after reading the following description, one skilled in the art could incorporate the quick pick mechanism into other types of tabbed browsers. As will be described in more detail below, the present quick pick mechanism provides a convenience that was unavailable until now.

Exemplary Computing Environment

[0019] The various embodiments of the quick pick mechanism may be implemented in different computer environments. The computer environment shown in **FIG. 1** is only one example of a computer environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computer environment be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in **FIG. 1**.

[0020] With reference to **FIG. 1**, one exemplary system for implementing a quick pick mechanism includes a computing device, such as computing device **100**. In a very basic configuration, computing device **100** typically includes at least one processing unit **102** and system memory **104**. Depending on the exact configuration and type of computing device, system memory **104** may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory **104** typically includes an operating system **106**, one or more program modules **108**, and may include program data **110**. This basic configuration is illustrated in **FIG. 1** by those components within dashed line **112**.

[0021] Computing device **100** may have additional features or functionality. For example, computing device **100** may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in **FIG. 1** by removable storage **114** and non-removable storage **116**. Computer storage media may include 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. System memory **104**, removable storage **114** and non-removable storage **116** are all examples of computer storage media. Thus, 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 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 computing device **100**. Any such computer storage media may be part of device **100**. Computing device **100** may also have input device(s) **118** such as keyboard **132**, mouse **134**, pen, voice input device, touch input device, etc. Output device(s) **120** such as a display, speakers, printer, etc. may also be included. These devices are well known in the art and need not be discussed at length here.

[0022] Computing device **100** may also contain communication connections **122** that allow the device to communicate with other computing devices **124**, such as over a network. Communication connection(s) **122** is one example of communication media. Communication media may typically be embodied by 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. The term computer readable media as used herein includes both storage media and communication media.

[0023] Various modules and techniques may be described herein in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. for performing particular tasks or implement particular abstract data types. These program modules and the like may be executed as native code or may be downloaded and executed, such as in a virtual machine or other just-in-time compilation execution environment. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0024] An implementation of these modules and techniques may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise “computer storage media” and “communications media.”

Exemplary Architecture

[0025] **FIG. 2** is a block diagram illustrating one embodiment of an architecture **200** for implementing the quick-pick mechanism as described herein. Architecture **200** includes one or more content windows **202**. The content window may be a web browser, a shell view, a word processing application/document, a spreadsheet application/document, a presentation application/document, a document viewer, or the like. Each content window **202** is associated with its own toolbars/menus **204**, custom bars **206**, status bars **208**, and content **210**, as needed. For example, custom bars may include bars created to operate with search engines and the like. As mentioned earlier, in traditional browsers, each browser window consumed space on a taskbar. Once a number of browser windows were opened, the taskbar became very cluttered. For the present tabbed browser, a tab window manager **220** handles all the content windows **202** via their respective tab window **212**. The tab window manager **220** communicates with a browser frame **214**. In addition, browser frame **214** communicates with a navigation bar **216**, a frame command bar **218**, and a tab user-interface **222**. Each of these components will be described below in more detail after describing thread boundary **230**.

[0026] Thread boundary **230** represents a boundary between a single thread (i.e., a frame thread) that handles the processing for the components shown to the left of thread boundary **230** and multiple threads that handle the processing for the components on the right of thread boundary **230**. Each instance of content window **202** is associated with at least one of its own threads. That thread also handles the processing for the associated toolbars/menus **204**, custom bars **206**, status bars **208**, and content **210**.

[0027] By having multiple threads, at least one for each content window **202**, the architecture prevents a potential bottleneck caused by having only one thread handle the messages for all HTML rendering across the multiple tabs. In addition, having multiple threads reduces the likelihood of unnecessary delays or timeouts when downloading web pages. Having multiple threads also makes the architecture more resilient because if a tab hangs, only one tab hangs instead of potentially hanging all of the tabs. This is in contrast with other tabbed browsers that host their tabs on a single UI thread. In these tabbed browsers, the entire application hangs when a single tab is blocked. Each individual thread may be prioritized.

[0028] Each of the components in architecture **200** is now described. As mentioned above, each content window **202** is associated with its own toolbars/menus **204**, custom bars **206**, status bars **208**, and content **210**, as needed. For example, if there are five tabs open, five instances of each add-on (e.g., custom bar, toolbar, etc) are created and hooked to their own content. Similarly, if menus or status bars are modified via internal logic, HTML, document hosting, add-ons, or the like, the menu or status bar associated with that content window **202** is changed. The add-ons and the content window **202** operate in a well known manner. Because the add-ons do not know about tabs, the architecture **200** does not modify the operation between the add-ons and the content window. Instead, the architecture wraps each set of content windows into one browser frame **214**. While not shown, there may be multiple browser frames **214**, each with a set of tabbed-heterogeneous windows.

[0029] Content window **202** maintains state for the content window, such as size, position, visibility of frame elements, and the like. The state may be persisted into a stream when the tabbed browser closes, and then may be read when a new content window is launched. Alternatively, the state may be saved in a registry and read from the registry when the new content window is launched.

[0030] The tab window **212** lives on the frame thread. Each tab window **212** is configured to manage communication between the outer frame components (e.g., navigation bar **216**, browser frame **214**, etc) and an individual tab. The tab window **212** holds pointers to its associated content window **202** and interfaces so that it can manage cross-thread communication between the outer frame components and the individual content window **202**. The cross-thread communication may involve cross-thread calls. In one embodiment, calls from the browser frame **214** to the content window **202** may be asynchronous. The tab window **212** is then responsible for marshalling the parameters and posting a message to the main window of content window **202**. In contrast, calls from the content window **202** to the frame browser **214** may be synchronous. The tab window **212** is then responsible for marshalling the interfaces of the corresponding tab window **212**. The tab window **212** may also convert synchronous calls into asynchronous calls in order to reduce the likelihood that a tab will hang.

[0031] The tab window **212** may also cache a limited amount of state. This allows the tab window to provide a quick synchronous lookup of state to other components. When the tab window **212** receives an event from the content window **202** that modifies the state, the tab window propagates these events up to the tab window manager **220**.

[0032] In addition, the tab window **212** exposes several operations that can be performed on a tab, such as destroy, set visibility, set size, set position, and order. The tab window **212** also exposes operations that the frame needs for asynchronously invoking a tab. These operations may be exposed via individual application programming interfaces (API) or through a generic API. For the present quick pick mechanism, each tab window **212** may have its own set of characteristics. For example, one of the tab windows may be a quick pick window which has a different navigation characteristic than other tabbed windows. As will be described and illustrated in **FIGS. 3-9**, the navigation characteristic for the quick pick window may modify a tab row and modify content displayed within a tabbed view of the tabbed browser.

[0033] The tab window manager **220** hosts multiple content windows **202** via multiple tab windows **212**. The tab window manager **220** manages the overall state of each of the tabs. The state may include: a top level URL; a page title; a back/forward button availability; a favorite icon for the current page, a progress state for the current page, security information reflected in the user-interface (e.g., HTTPs info); and the like. The tab window manager **220** may synchronously query the tab window **212** for per-tab data. In addition, the tab window manager **220** manages the operations performed on the tabs, such as open, close, select, move, set size, set position, set order, and the like. This is achieved by accessing the operations exposed via the tab window **212**.

[0034] The browser frame **214** hosts the navigation bar **216**, frame command bar **218**, tab window manager **220**, and the tab UI **222**. The browser frame **214** may also directly or indirectly host a rebar control (not shown). The browser frame **214** brokers communication between these hosted components.

[0035] The tab UI **222** is configured to render the user-interface for the collection of tab windows **212**. As will be described later in conjunction with **FIGS. 3-9**, the Tab UI **222** may provide a quick pick user-interface for implementing the functionality for the present quick pick mechanism. In addition, the tab UI **222** controls the rectangle for the tabs' browser region. Tab UI **222** receives events from tab window manager **220** via browser frame **214**. The events pertain to the tabs being managed by tab window manager **220**. The events include an "OnTabListChanged" event, an "OnTabChanged" event, and the like. The "OnTabListChanged" event is sent by tab window manager **220** when changes are made to a collection of tabs (e.g., Open, Close, Reorder, etc). The "OnTabChanged" event is sent by tab window manager **220** when a single tab changes (e.g., progress, title, URL). The corresponding tab window **212** sends a notification to tab window manager **220** which forwards the notification to tab UI **222**. The notification identifies the tab. This method of communication keeps the communication serialized. In an alternate embodiment, individual tab windows **212** may directly call into tab UI **222**. In addition, there may be multiple tab UI **222** components, each one handling the same set of tabs, but for a different location on the user-interface. For example, one tab UI may handle a tab row along the top of the display and another tab UI may handle a tab listview along the side of the display.

[0036] The frame command bar **218** provides a set of commands that have been determined to be the most useful.

By having this set of commands readily available for any tab, a user can conveniently locate one of the commands for any tab.

Exemplary User-Interface

[0037] **FIG. 3** is a display illustrating one embodiment of a user-interface that supports the quick pick mechanism in a tabbed browser. User-interface **300** illustrates salient portions of a tabbed browser that implements the present quick pick mechanism. Those skilled in the art will appreciate that optional third party toolbars (not shown), a page space control (not shown), a status bar (not shown) and/or custom bars (not shown) may also be included within user-interface **300** without departing from the present quick pick mechanism. As shown, user-interface **300** includes a navigation bar **302** and a tabbed window **312**.

[0038] The tabbed window **312** includes, a tab band or tab row **320**, one or more tabs (e.g., tabs **332-342**), and content **360** associated with the tab currently in focus (e.g., tab **336** shown in **FIG. 3**). In addition, tabbed window **312** may include a new tab **356** for creating a new tab within the tabbed window **312**. As additional tabs are opened, the visible portion of each tab is reduced to accommodate the additional tabs. Once there is inefficient space for an additional tab, some of the tabs are visually removed from the tabbed window **312**. However, visually removed tabs are still maintained as tab windows (tab window **212**) and are managed by the tab window manager **220** shown in **FIG. 2**. The visually removed tabs may be accessed by clicking either a left or right overflow indicator **352** and **354**, respectively. The left and right overflow indicator **352** and **354** appear within the tabbed window **312** when one or more of the open tabs are visually removed from the tab row **320**. When there are a limited number of open tabs (e.g., four tabs), the text within each tab may be sufficient to distinguish between the tabs, however once a user has opened several tabs, it is difficult to distinguish among the several open tabs because the text associated with each tab is reduced and some of the tabs may not even be visible. For illustrative purposes, the present example displays the text for the title of each tab using its original order (e.g., 1st) appended with "Site". As one can see, as more tabs are opened, titles for certain tabs (e.g., tabs **340** and **341**) display limited useful information about the content of the associated open tab.

[0039] In accordance with the present quick pick mechanism, the tabbed browser provides a mechanism for accessing a quick pick user interface. The mechanism may be a quick pick button **350**, a context menu selection (not shown), and/or the like on the user-interface **300**. The mechanism may also be a hot key (not shown) entered through a keyboard. A user may select the quick pick button **350** to access the quick pick user interface.

[0040] The navigation bar **302** may include an address bar **304**, a back button **306**, a forward button **308**, and a search box **310**. The content of the address bar **304** may be maintained for each tab. Thus, when "clicking" through the tabs **332-341** the address bar **304** may display the address for the currently selected tab (i.e., the tab in focus). The search box **310** allows text to be entered and searched for in the currently selected tab.

[0041] In one embodiment, items in the user-interface **300** that are per frame include the navigation bar **302** and the tab

band **320**. Items in the user-interface **300** that are per tab include content **360**. In one embodiment, content **360** may include different type of data sources. For example, content **360** may be a web page, a shell folder, a navigation based application, and the like. This heterogeneous content may be hosted in a single frame. In another embodiment, the content **360** associated with each tab may all be web pages.

[0042] **FIG. 4** is a display illustrating one embodiment for the quick pick user-interface and illustrating a user's selection within the interface. Quick pick user-interface **400** illustrates navigation bar **302** as described above and a quick pick window **460**. The quick pick window **460** displays a rich set information, such as a graphical representation (e.g., thumbnails **402-411**) for each of the open tabs in the tab row **320**, a graphical representation (e.g., thumbnails **401**, **412-414**) of the open tabs that are not currently visible in the tab row because they are in the overflow. The tabbed browser scales the thumbnails so that the content of each tab can be viewed in the quick pick window **460**. In the example, thumbnails **401-405** are displayed consecutively on a top row, thumbnails **406-410** are displayed consecutively on a middle row, and thumbnails **411-414** are displayed consecutively on a bottom row. The thumbnails for each of the open tabs are readily available in memory, thus displaying quick pick window **460** provides an efficient means for distinguishing the open tabs. In certain situations, if the user has opened a significant number of tabs, the quick pick window **460** may display further scaled down thumbnails, a predetermined number of tabs, or the like.

[0043] Each thumbnail may include a close indicator (e.g., close indicator **420**). If a user selects close indicator **420**, the corresponding thumbnail (e.g., thumbnail **411**) is removed from the quick pick window **460** and its associated tab (e.g., tab **341**) is removed from the tab row **320**. In addition, the rich set of information may include a page title or other adornment for each item in the quick pick window **460**. When a user hovers a pointer (e.g., pointer **422**) over one of the thumbnails (e.g., thumbnail **410**), the thumbnail **410** may visually change appearance, such as changing its color. In addition, the tab (tab **440**) associated with the hovered thumbnail **410** may change its appearance, such as changing color, bolding text, and the like. The user may select any one of the thumbnails to view its corresponding content in the tabbed window.

[0044] **FIG. 5** is a display illustrating the tabbed window after the user's action within the quick pick user-interface shown in **FIG. 4**. In this example, the user selected thumbnail **410** in the quick pick user-interface shown in **FIG. 4**. Thus, the tabbed window displays the tab (i.e., tab **340**) associated with thumbnail **410** as the in focus tab. Content of the in focus tab is displayed as the content **560** in the tabbed window **312**.

[0045] **FIG. 6** is a display illustrating the quick pick user-interface and illustrating another user's selection within the interface. This display is similar to **FIG. 4** except that in **FIG. 6** the user selects thumbnail **413**. Because the tab associated with thumbnail **413** is not currently visible in the tab row **320**, the right overflow indicator **354** may visually change appearance so that the user may easily identify the relative location for the associated tab in the tab row **320**. In another embodiment, when the user hovers over a thumbnail (e.g., thumbnail **413**), the thumbnail may enlarge so that the

user can more easily view the content associated with the open tab. A page title (e.g., page title 630) or other adornment may be associated with each thumbnail. This provides additional information to the user in helping the user select one of the open tabs.

[0046] FIG. 7 is a display illustrating the tabbed window after the user's action within the quick pick user-interface shown in FIG. 6. As described above, the user selected thumbnail 413 in the quick pick user-interface. Thus, the tabbed window displays the tab (i.e., tab 343) associated with thumbnail 413 as the in focus tab. Content of the in focus tab is displayed as the content 760 in the tabbed window 312. It is important to note that for this selection, the tabs displayed within the tab row 320 have been re-positioned so that the selected tab (i.e., tab 343) is in a favorable position within the visible tabs. Thus, tabs 332-334 (shown in FIG. 5) have been visually removed and are now accessible by clicking on the left overflow indicator 352. Likewise, tabs 342-344 which were previously in the overflow in FIG. 5, now appear in the tab row 320. In essence, the tabbed browser attempts to intelligently slide the tabs in the tab row based on the item selected in the quick pick user interface.

[0047] FIG. 8 is a display illustrating the quick pick user-interface and illustrating a user re-positioning a thumbnail within the interface. Again this display is identical to FIG. 4 except that in FIG. 8 the user is performing a drag-drop operation within the quick pick user interface. Thumbnail 413 is being re-positioned from the second to last position in the bottom row to the third position in the top row. After the drag-drop operation, thumbnail 413 will be between thumbnail 402 and thumbnail 403 on the top row.

[0048] FIG. 9 is a display illustrating the tabbed window after the user's action within the quick pick user-interface shown in FIG. 8. In FIG. 8, after moving thumbnail 413, the user selected thumbnail 413 for viewing. As one can see, the tabs within the tab row 320 shown in FIG. 9 have been re-positioned so that the tab (i.e., tab 343) associated with thumbnail 413 is positioned between tab 332 and 333 associated with thumbnails 402 and 403, respectively. In addition, tab 331 which was previously in the overflow has been included in the tab row 320 and tabs 440 and 441 have been removed from the tab row and are now in the overflow. This allows the selected tab 343 to have a more favorable position than if tab 331 remained in the overflow. In one embodiment, the most favorable position may be the center tab in the tab row. Alternatively, the most favorable position may be the left most tab in the tab row. This most favorable position may be user-definable.

Exemplary Operation

[0049] FIG. 10 is an operational state diagram 1000 generally illustrating the processing for managing and selecting one out of a large set of open tabs within a tabbed browser using the present quick pick feature. The processing may be implemented on a computing device, such as computing device 100. The processing may be implemented within operating system 106, as one or more program modules 108, or in a combination of the two. For illustration purposes, the processing is illustrated and described as different transitions from different states. However, several flow diagrams may have been used to illustrate all the processing occurring within state diagram 1000. At state

1002, the tabbed browser displays a tabbed window that includes a tab row and contents of the currently in focus tab. From state 1002, several transitions may occur.

[0050] State 1004 may occur from state 1002. At state 1004, the tabbed browser adds a new tab and changes the in focus tab to the new tab. In addition, the tabbed browser may visually remove some of the tabs displayed on the tab row if the addition of the new tab makes the number of tabs in the tab row more than a pre-determined number. After the new tab is added, the state transitions back to state 1002.

[0051] State 1006 may occur from state 1002. At state 1006, the tabbed browser responds to the user's selection of a tab or overflow indicator. The tabbed browser adjusts the in focus tab and adjusts the tabs that are displayed on the tab row accordingly. After the tabbed browser handles the user's selection, the state transitions back to state 1002.

[0052] State 1008 may occur from state 1002. State 1008 is responsible for handling the present quick pick mechanism. From state 1008, several transitions may occur.

[0053] State 1010 may occur from state 1008. State 1010 exits the quick pick mechanism, such as via a close button on the quick pick user interface. From state 1008, the tabbed browser transitions back to state 1002.

[0054] State 1012 may occur from state 1008. At state 1012, the tabbed browser closes a thumbnail in response to a user's action within the quick pick user interface. For example, the user may have selected a close button 420 within the quick pick user interface. Upon closing the thumbnail, the tabbed browser may also perform other operations associated with closing the thumbnail, such as removing the associated tab from the tab row, deleting the associated tab window, and informing the tab window manager that the associated tab window is no longer open. After the tabbed browser handles closing the thumbnail, the state transitions back to state 1008.

[0055] State 1014 may occur from state 1008. At state 1014, the tabbed browser may rearrange the thumbnails in response to a user's action within the quick pick user interface. For example, the user may have selected one of the thumbnails, dragged it to another location, and dropped it next to different thumbnails. The tab browser will change the order of the thumbnails displayed in the quick pick user interface. In addition, the tab browser will change the order of the tabs within the tab row to reflect the re-positioned thumbnail. After the tabbed browser handles the re-arranging of the thumbnails, the state transitions back to state 1008.

[0056] State 1016 may occur from state 1008. At state 1016, the tabbed browser may perform operations in response to the user's selection of one of the thumbnails displayed in the quick pick user interface. For example, the tabbed browser may set the in focus tab to be the tab associated with the selected thumbnail. In addition, the tabbed browser may intelligently slide the tabs within the tab row so that the selected thumbnail is displayed in a favorable position. For example, if the selected thumbnail corresponds to a tab in the overflow on the left side of the tab row, each tab in the tab row may move to the right until there are not any more items in the overflow on the left. Alternatively, each tab in the tab row may move to the right until the selected item moves to a favorable position on the tab row, such as in the center. Likewise, if the selected thumbnail

corresponds to a tab in the overflow on the right side of the tab row, each tab in the tab row may move to the left until there are not any more items in the overflow on the right. Alternatively, each tab in the tab row may move to the left until the selected item moves to a favorable position on the tab row. The favorable position may be user definable. Once the tabbed browser handles the processing based on the selected thumbnail, the state transitions to back to state 1002.

[0057] As one can see, the present quick pick mechanism allows a user to conveniently select any one of several open tabs. They can easily view each open tab, rearrange the tabs, close a tab, and the like. Thus, the present quick pick mechanism provides an enhanced and convenient viewing experience to users.

[0058] Reference has been made throughout this specification to “one embodiment,” “an embodiment,” or “an example embodiment” meaning that a particular described feature, structure, or characteristic is included in at least one embodiment of the present invention. Thus, usage of such phrases may refer to more than just one embodiment. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0059] One skilled in the relevant art may recognize, however, that the present mechanism may be practiced without one or more of the specific details, or with other methods, resources, materials, etc. In other instances, well known structures, resources, or operations have not been shown or described in detail merely to avoid obscuring aspects of the present consistent visual appearance technique.

[0060] While example embodiments and applications have been illustrated and described, it is to be understood that the present technique is not limited to the precise configuration and resources described above. Various modifications, changes, and variations apparent to those skilled in the art may be made in the arrangement, operation, and details of technique disclosed herein without departing from the scope of the claimed invention.

1. A computer-implemented method for navigating content, the method comprising:

displaying a tabbed browser window on a display, the tabbed browser window displaying a plurality of open tabs, each open tab having content associated therewith;

displaying a window on the display in response to a user action, the window displaying a set of graphic views, each graphic view being associated with one of the plurality of open tabs and displaying at least a portion of the content associated with that open tab;

receiving another user action; and

modifying the display based on the other user action.

2. The computer-implemented method of claim 1, wherein the user action comprises a selection of a display element displayed in the tabbed browser window.

3. The computer-implemented method of claim 1, wherein each graphical view comprises a thumbnail displaying the portion of content.

4. The computer-implemented method of claim 1, wherein the other user action comprises selection of a close indicator associated with one of the graphical views and modifying the display comprises removing the one graphical view from being displayed in the window.

5. The computer-implemented method of claim 4, wherein modifying the display further comprises removing the open tab associated with graphical view from being displayed as one of the plurality of open tabs in the tabbed browser window.

6. The computer-implemented method of claim 1, wherein the other user action comprises a drag and drop operation with one of the graphical views and modifying the display comprises re-positioning the one graphical view based on where the graphical view is dropped within the window and adjusting the open tabs displayed in the tabbed browser window to re-position the open tab associated with the one graphical view accordingly.

7. The computer-implemented method of claim 1, wherein the other user action comprises a selection of one of the graphical views and modifying the display comprises setting an in focus tab in the tabbed browser window to be the open tab associated with the selected graphical view.

8. The computer-implemented method of claim 1, wherein the other user action comprises hovering over one of the graphical views and modifying the display comprises visually changing the appearance of the tab in the tabbed browser window associated with the one graphical view.

9. The computer-implemented method of claim 1, wherein the other user action comprises hovering over one of the graphical views and modifying the display comprises visually enlarging the one graphical view.

10. The computer-implemented method of claim 1, wherein the other user action comprises selecting one of the graphical views and modifying the display comprises switching back to the tabbed browser window in which the content of the open tab associated with the graphical view is displayed and the open tab associated with the graphical view is displayed in a favorable position with respect to the other open tabs.

11. A computer-readable medium having computer-executable instructions for selecting open tabs within a tabbed-browsing software, the instructions comprising:

displaying a quick pick user interface in response to a user action, the quick pick user interface displaying a plurality of graphical views, each graphical view corresponding to one of a plurality of open tabs within the tabbed browsing software and displaying at least a portion of content associated with the open tab;

receiving another user action; and

modifying the quick pick user interface based on the other user action.

12. The computer-readable medium of claim 11, wherein the user action comprises a selection of a quick pick icon displayed in a tabbed window view of the tabbed browsing software.

13. The computer-readable medium of claim 11, wherein each graphical view comprises a thumbnail, the thumbnail displaying a screen shot of content associated with the corresponding open tab.

14. The computer-readable medium of claim 11, wherein the other user action comprises a selection of one of the graphical views and modifying the quick pick user interface

comprises setting an in focus tab to be the open tab associated with the selected graphical view.

15. The computer-readable medium of claim 11, wherein the other user action comprises hovering over one of the graphical views and modifying the quick pick user interface comprises visually changing the appearance of the associated tab.

16. The computer-readable medium of claim 11, wherein the other user action comprises hovering over one of the graphical views and modifying the quick pick user interface comprises visually enlarging the one graphical view.

17. The computer-readable medium of claim 11, wherein the other user action comprises selecting one of the graphical views and modifying the quick pick user interface comprises switching back to a tabbed view in which the content of the open tab associated with the graphical view is displayed and the open tab associated with the graphical view is displayed in a favorable position within a tab row.

18. A system configured to select one of several open tabs within a tabbed browser, the system comprising:

a processor; and

a memory into which a plurality of computer-executable instructions are loaded, the plurality of instructions performing a method comprising:

displaying a quick pick user interface in response to a user action, the quick pick user interface displaying a plurality of graphical views, each graphical view corresponding to one of a plurality of open tabs within the tabbed browsing software and displaying at least a portion of content associated with the open tab;

receiving another user action; and

modifying the quick pick user interface based on the other user action.

19. The system of claim 18, wherein each graphical view comprises a thumbnail, the thumbnail displaying a screen shot of content associated with the corresponding open tab

20. The system of claim 18, wherein the other user action comprises selecting one of the graphical views and modifying the quick pick user interface comprises switching back to a tabbed view in which the content of the open tab associated with the graphical view is displayed and the open tab associated with the graphical view is displayed in a favorable position within a tab row.

21. The computer-implemented method of claim 1, wherein the user action comprises a selection of a menu item within a context menu associated with the tabbed browser window.

22. The computer-implemented method of claim 1, wherein the user action comprises a selection of a hot key entered via a keyboard.

23. The computer-implemented method of claim 1, wherein the graphical view comprises a screen shot of the portion of content.

24. The computer-implemented method of claim 23, further comprising accessing the screen shot from memory when displaying the graphical view in the window.

25. The computer-implemented method of claim 8, wherein visually changing the appearance of the tab comprises changing a color of the tab displayed in the tabbed browser window.

26. The computer-implemented method of claim 3, further comprising scaling at least one thumbnail to allow each thumbnail to be viewed in the window.

27. The computer-implemented method of claim 1, wherein the set of graphic views comprises a pre-determined number.

28. The computer-implemented method of claim 10, wherein the favorable position is user-definable.

29. The computer-implemented method of claim 1, wherein the at least a portion of the content comprises the entire content associated with the one open tab.

* * * * *