Title: SINGLE INPUT GRAPHICAL USER INTERFACE CONTROL ELEMENT AND METHOD

Abstract: Described are methods, systems and computer readable media for GUI control elements and associated processing methods.
SINGLE INPUT GRAPHICAL USER INTERFACE CONTROL ELEMENT AND
METHOD

[0001] This application claims the benefit of U.S. Provisional Application No. 62/161,813, entitled "Computer Data System" and filed on May 14, 2015, which is incorporated herein by reference in its entirety.

[0002] Embodiments relate generally to computer user interfaces, and more particularly, to methods, systems and computer readable media for graphical user interface (GUI) control elements and associated methods.

[0003] Some graphical user interfaces may provide multiple views, tabs, widgets or other GUI elements within a display view such as a window. The multiple elements may not be subject to control from one or more GUI elements. Thus, a user may have to manually update or modify each GUI element within the display view in order to change the information presented in each view. Such a manual update or change process may be time consuming and/or error prone. Moreover, the multiple elements may not be linked to one or more GUI control elements configured to cause the individual elements to display updated information retrieved from a data source in response to a signal from the control element.

[0004] Embodiments were conceived in light of the above mentioned needs, problems and/or limitations, among other things.

[0005] Some implementations can include a method of controlling a graphical user interface (GUI) having a plurality of elements based on input from a single GUI control element. The method can include providing a view tree containing nodes corresponding to graphical user interface (GUI) elements within a parent window, and determining presence of a first GUI control element in the view tree of the parent window. The method can also include receiving an event from the first GUI control element, the event corresponding to a single input event to the GUI control element, and in response to the received event, traversing the view tree. The method can further include at each node of the view tree reached during the traversing: sending a notification signal of the event, when a node has listening for events from the first GUI control element enabled, updating the GUI element associated with the node based on the GUI control element notification signal, and causing the updated GUI element to be displayed.

[0006] The GUI element can include one of a view and a GUI element. The method can also include disabling listening for a notification signal from the first GUI control element at a
node when an indication is received to disable GUI control element listening for that node. The method can further include updating the view tree when a GUI element is added to, removed from, or moved within the parent window or removed from the parent window.

[0007] The method can also include providing a second GUI control element within the parent window, the second GUI control element configured to listen for notifications from the first GUI control element. The view tree can include a partial view tree.

[0008] Some implementations can include a system for controlling a graphical user interface (GUI) having a plurality of elements based on input from a single GUI control element, the system comprising one or more hardware processors coupled to a nontransitory computer readable medium having stored thereon software instructions that, when executed by the one or more processors, cause the one or more processors to perform operations. The operations can include providing a view tree containing nodes corresponding to graphical user interface (GUI) elements within a parent window, and determining presence of a first GUI control element in the view tree of the parent window. The operations can also include receiving an event from the first GUI control element, the event corresponding to a single input event to the GUI control element, and in response to the received event, traversing the view tree. The operations can further include, at each node of the view tree reached during the traversing, sending a notification signal of the event, when a node has listening for events from the first GUI control element enabled, updating the GUI element associated with the node based on the GUI control element notification signal, and causing the updated GUI element to be displayed.

[0009] The GUI element can include one of a view and a GUI element. The operations can further include disabling listening for a notification signal from the first GUI control element at a node when an indication is received to disable GUI control element listening for that node. The operations can also include updating the view tree when a GUI element is added to, removed from, or moved within the parent window or removed from the parent window.

[0010] The operations can further include providing a second GUI control element within the parent window, the second GUI control element configured to listen for notifications from the first GUI control element. The view tree can be a subtree.

[0011] Some implementations can include a nontransitory computer readable medium having stored thereon software instructions that, when executed by one or more processors, cause the one or more processors to perform operations. The operations can include providing a view tree containing nodes corresponding to graphical user interface (GUI) elements within a parent window, and determining presence of a first GUI control element in the view tree of
the parent window. The operations can also include receiving an event from the first GUI control element, the event corresponding to a single input event to the GUI control element, and in response to the received event, traversing the view tree. The operations can further include, at each node of the view tree reached during the traversing, sending a notification signal of the event, when a node has listening for events from the first GUI control element enabled, updating the GUI element associated with the node based on the GUI control element notification signal, and causing the updated GUI element to be displayed.

[0012] The GUI element can include one of a view and a GUI element. The operations can further include disabling listening for a notification signal from the first GUI control element at a node when an indication is received to disable GUI control element listening for that node. The operations can also include updating the view tree when a GUI element is added to, removed from, or moved within the parent window or removed from the parent window.

[0013] The operations can further include providing a second GUI control element within the parent window, the second GUI control element configured to listen for notifications from the first GUI control element. The view tree can be a subtree.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIG. 1 is a diagram of an example computer data system showing an example data distribution configuration in accordance with some implementations.

[0015] FIG. 2 is a diagram of an example computer data system showing an example administration/process control arrangement in accordance with some implementations.

[0016] FIG. 3 is a diagram of an example computing device configured for GUI control element processing in accordance with some implementations.

[0017] FIG. 4 is a diagram of an example GUI with a GUI control element in accordance with some implementations.

[0018] FIG. 5 is a diagram of an example GUI with a plurality of GUI control elements in accordance with some implementations.

[0019] FIG. 6 is a flowchart showing a GUI control element method in accordance with some implementations.

DETAILED DESCRIPTION

[0020] Reference may be made herein to the Java programming language, Java classes, Java bytecode and the Java Virtual Machine (JVM) for purposes of illustrating example
implementations. It will be appreciated that implementations can include other programming languages (e.g., groovy, Scala, R, Go, etc.), other programming language structures as an alternative to or in addition to Java classes (e.g., other language classes, objects, data structures, program units, code portions, script portions, etc.), other types of bytecode, object code and/or executable code, and/or other virtual machines or hardware implemented machines configured to execute a data system query.

[0021] FIG. 1 is a diagram of an example computer data system and network 100 showing an example data distribution configuration in accordance with some implementations. In particular, the system 100 includes an application host 102, a periodic data import host 104, a query server host 106, a long-term file server 108, and a user data import host 110. While tables are used as an example data object in the description below, it will be appreciated that the data system described herein can also process other data objects such as mathematical objects (e.g., a singular value decomposition of values in a given range of one or more rows and columns of a table), TableMap objects, etc. A TableMap object provides the ability to lookup a Table by some key. This key represents a unique value (or unique tuple of values) from the columns aggregated on in a byExternalQ statement execution, for example. A TableMap object can be the result of a byExternalQ statement executed as part of a query. It will also be appreciated that the configurations shown in FIGS. 1 and 2 are for illustration purposes and in a given implementation each data pool (or data store) may be directly attached or may be managed by a file server.

[0022] The application host 102 can include one or more application processes 112, one or more log files 114 (e.g., sequential, row-oriented log files), one or more data log tailers 116 and a multicast key-value publisher 118. The periodic data import host 104 can include a local table data server, direct or remote connection to a periodic table data store 122 (e.g., a column-oriented table data store) and a data import server 120. The query server host 106 can include a multicast key-value subscriber 126, a performance table logger 128, local table data store 130 and one or more remote query processors (132, 134) each accessing one or more respective tables (136, 138). The long-term file server 108 can include a long-term data store 140. The user data import host 110 can include a remote user table server 142 and a user table data store 144. Row-oriented log files and column-oriented table data stores are discussed herein for illustration purposes and are not intended to be limiting. It will be appreciated that log files and/or data stores may be configured in other ways. In general, any data stores discussed herein could be configured in a manner suitable for a contemplated implementation.
In operation, the input data application process 112 can be configured to receive input data from a source (e.g., a securities trading data source), apply schema-specified, generated code to format the logged data as it's being prepared for output to the log file 114 and store the received data in the sequential, row-oriented log file 114 via an optional data logging process. In some implementations, the data logging process can include a daemon, or background process task, that is configured to log raw input data received from the application process 112 to the sequential, row-oriented log files on disk and/or a shared memory queue (e.g., for sending data to the multicast publisher 118). Logging raw input data to log files can additionally serve to provide a backup copy of data that can be used in the event that downstream processing of the input data is halted or interrupted or otherwise becomes unreliable.

A data log tailer 116 can be configured to access the sequential, row-oriented log file(s) 114 to retrieve input data logged by the data logging process. In some implementations, the data log tailer 116 can be configured to perform strict byte reading and transmission (e.g., to the data import server 120). The data import server 120 can be configured to store the input data into one or more corresponding data stores such as the periodic table data store 122 in a column-oriented configuration. The periodic table data store 122 can be used to store data that is being received within a time period (e.g., a minute, an hour, a day, etc.) and which may be later processed and stored in a data store of the long-term file server 108. For example, the periodic table data store 122 can include a plurality of data servers configured to store periodic securities trading data according to one or more characteristics of the data (e.g., a data value such as security symbol, the data source such as a given trading exchange, etc.).

The data import server 120 can be configured to receive and store data into the periodic table data store 122 in such a way as to provide a consistent data presentation to other parts of the system. Providing/ensuring consistent data in this context can include, for example, recording logged data on a disk or memory, ensuring rows presented externally are available for consistent reading (e.g., to help ensure that if the system has part of a record, the system has all of the record without any errors), and preserving the order of records from a given data source. If data is presented to clients, such as a remote query processor (132, 134), then the data may be persisted in some fashion (e.g., written to disk).

The local table data server 124 can be configured to retrieve data stored in the periodic table data store 122 and provide the retrieved data to one or more remote query processors (132, 134) via an optional proxy.
[0027] The remote user table server (RUTS) 142 can include a centralized consistent data writer, as well as a data server that provides processors with consistent access to the data that it is responsible for managing. For example, users can provide input to the system by writing table data that is then consumed by query processors.

[0028] The remote query processors (132, 134) can use data from the data import server 120, local table data server 124 and/or from the long-term file server 108 to perform queries. The remote query processors (132, 134) can also receive data from the multicast key-value subscriber 126, which receives data from the multicast key-value publisher 118 in the application host 102. The performance table logger 128 can log performance information about each remote query processor and its respective queries into a local table data store 130. Further, the remote query processors can also read data from the RUTS, from local table data written by the performance logger, or from user table data read over NFS, for example.

[0029] It will be appreciated that the configuration shown in FIG. 1 is a typical example configuration that may be somewhat idealized for illustration purposes. An actual configuration may include one or more of each server and/or host type. The hosts/servers shown in FIG. 1 (e.g., 102-1 10, 120, 124 and 142) may each be separate or two or more servers may be combined into one or more combined server systems. Data stores can include local/remote, shared/isolated and/or redundant. Any table data may flow through optional proxies indicated by an asterisk on certain connections to the remote query processors. Also, it will be appreciated that the term "periodic" is being used for illustration purposes and can include, but is not limited to, data that has been received within a given time period (e.g., millisecond, second, minute, hour, day, week, month, year, etc.) and which has not yet been stored to a long-term data store (e.g., 140).

[0030] FIG. 2 is a diagram of an example computer data system 200 showing an example administration/process control arrangement in accordance with some implementations. The system 200 includes a production client host 202, a controller host 204, a GUI host or workstation 206, and query server hosts 208 and 210. It will be appreciated that there may be one or more of each of 202-210 in a given implementation.

[0031] The production client host 202 can include a batch query application 212 (e.g., a query that is executed from a command line interface or the like) and a real time query data consumer process 214 (e.g., an application that connects to and listens to tables created from the execution of a separate query). The batch query application 212 and the real time query data consumer 214 can connect to a remote query dispatcher 222 and one or more remote query processors (224, 226) within the query server host 1 208.
[0032] The controller host 204 can include a persistent query controller 216 configured to connect to a remote query dispatcher 232 and one or more remote query processors 228-230. In some implementations, the persistent query controller 216 can serve as the "primary client" for persistent queries and can request remote query processors from dispatchers, and send instructions to start persistent queries. For example, a user can submit a query to 216, and 216 starts and runs the query every day. In another example, a securities trading strategy could be a persistent query. The persistent query controller can start the trading strategy query every morning before the market opened, for instance. It will be appreciated that 216 can work on times other than days. In some implementations, the controller may require its own clients to request that queries be started, stopped, etc. This can be done manually, or by scheduled (e.g., cron) jobs. Some implementations can include "advanced scheduling" (e.g., auto-start/stop/restart, time-based repeat, etc.) within the controller.

[0033] The GUI/host workstation can include a user console 218 and a user query application 220. The user console 218 can be configured to connect to the persistent query controller 216. The user query application 220 can be configured to connect to one or more remote query dispatchers (e.g., 232) and one or more remote query processors (228, 230).

[0034] FIG. 3 is a diagram of an example computing device 300 in accordance with at least one implementation. The computing device 300 includes one or more processors 302, operating system 304, computer readable medium 306 and network interface 308. The memory 306 can include a GUI control element (e.g., a single input GUI control element or a "One Click" GUI control element) application 310 and a data section 312 (e.g., for storing view trees, etc.).

[0035] In operation, the processor 302 may execute the application 310 stored in the memory 306. The application 310 can include software instructions that, when executed by the processor, cause the processor to perform operations for remote data table publishing/subscribing using a multicast key-value protocol in accordance with the present disclosure (e.g., performing one or more of 602-616 described below).

[0036] The application program 310 can operate in conjunction with the data section 312 and the operating system 304.

[0037] In general, some implementations provide an advantage by letting the user re-configure a GUI on the fly, with no coding. The user can re-configure the GUI and have the filtering continue to work with little or no assistance from programmers or other information technology personnel needed.
[0038] Regarding FIGS. 4-6, it will be appreciated that windows are being used for illustration purposes and that other types of user interface elements could be used in place of windows. The terms single input GUI control element, "single input", "one click GUI control element", "one click" and/or "one click GUI element" are used herein for purposes of illustration and to convey the notion that single GUI control element in combination with a single input event (e.g., "one click") to that GUI control element can control potentially multiple views/GUI elements within a parent window with "one click" or one input by the user. It should be appreciated that the term "click" is being used as an example user interface input mode and that the GUI control element could receive one or more other user inputs in one or more modes such as mouse clicks, mouse movements, mouse drags, mouse focus, mouse hover, touch screen gestures (selections, press, long press, taps, drags, or other gestures) or other haptic input, keyboard commands, voice commands, visual gestures received by an imaging device, or the like. A single input event can include a series of keystrokes followed by a final keystroke or other final GUI input indicating completion of the typing or character entry (e.g., an enter key or other final input keystroke or GUI input). In general, any input event detectable by a computer could be used to activate the one click GUI control element to cause an update to the views and/or GUI elements within the same parent window as the one click GUI control element and which have the one click GUI control feature enabled as described herein. A parent window can include zero or more views, zero or more other GUI elements (e.g., leaf nodes such as a table view, a plot, etc. and interior nodes such as a frame (or main window), internal frame, tabbed pane, etc.) and zero or more one click GUI control elements. Some implementations can include widget views that may be leaves. Widget views can be something as simple as a text symbol and information such as price, status, day-on-day change, etc. in a convenient layout. Or they could be more complex elements e.g., aggregating positions by account in a tree view; or multiple tables organizing options views according to expiration. In some implementations, interior nodes can include split panes, tabbed panes, and root windows. Elements can also include a popout.

[0039] Some implementations can traverse a subtree of a full view tree discussed herein. For example, when a one click GUI control element traverses the tree, it finds its immediate parent and traverses the subtree defined by that parent. An implementation could traverse the full tree as well.

[0040] Furthermore, when possible values are being determined for the one click GUI control element (e.g. all stock symbols), just the subtree may be traversed.
[0041] Also, some implementations may traverse a tree that is rooted at an ancestor window, which is not necessarily the direct parent of the one click element (it is the first window). A system can traverse any arbitrary subtree (e.g., being more expansive to include other windows, or being more fine grained to not include elements beyond a parent).

[0042] The one click GUI control elements may not determine the possible values by traversing any trees. For example, a user could either type in an arbitrary value and it matches (or not). Some one click GUI control elements have a list of inputs that you can click through, which may be input by the user, or obtained by the system from a data object, or via some other automated data-driven method.

[0043] FIG. 4 is a diagram of an example user interface 400 having a first parent window 402 and a second parent window 404. The user interface 400 may be displayed on a display device connected to a GUI/host workstation (e.g., 206). The first parent window 402 includes a view tree 403 having a first view 406 (for example a tabbed panel), a one click GUI control element 408, a second view 410, a first GUI element 412 and a second GUI element 414. The view tree for the first parent window 402 is created and/or modified when view or GUI elements are placed (or activated) within the first parent window 402 or removed from the first parent window 402. The second parent window 404 includes a view tree having a third view 416, a third GUI element 418 and a fourth GUI element 420. It will be appreciated that the view trees are shown within the parent windows for illustration purposes. In an actual implementation, the views and other GUI elements would be displayed in the parent windows for interaction with a user and the view trees would be logical data structures residing in data storage (e.g., RAM or disk).

[0044] In operation, when a "click" or similar GUI event (e.g., touch screen selection, keyboard code, voice command, visual gesture, etc.) occurs on the one click GUI control element 408, the view tree for the parent window containing the one click GUI control element is traversed. In this example, the view tree for the first parent window 402 is traversed because the one click GUI control element 408 is within the first parent window 402. As the view tree 403 is traversed, a notification is provided to each view or GUI element (e.g., 410, 412 and 414) in the view tree informing the view or GUI element of the "click" event on the one click GUI control element. In response to the notification, each view or other GUI element that is configured to respond to one click GUI control element notifications (e.g., does not have the one click GUI control excluded) updates its respective view or GUI element based on the one click GUI control element click event, which can
include the information for filtering (or other operations) to cause the GUI elements listening to the one click GUI control element to update.

[0045] FIG. 5 shows a diagram of an example parent window 502 having a view tree 503 with a first view 504, a first one click GUI control element 506, a second view 508, a first GUI element 510 (one click disabled), a second GUI element 512 and a second one click GUI control element 514. The user interface 500 may be displayed on a display device connected to a GUI/host workstation (e.g., 206).

[0046] In an operational example, a user makes a selection or "clicks" the first one click GUI control element 506. This event causes the system to traverse the view tree 503 of the parent window 502. Accordingly, the first view 504, the second view 508, and the second GUI element 512 may be updated in response to the notification of the click event from the first one click GUI control element 506.

[0047] The first GUI element 510 has the one click GUI control feature disabled. Accordingly, the first GUI element 510 will not update in response to the notification signal from the first one click GUI control element 506. For example, an implementation can provide a right-click pulldown menu on the GUI component that has a "Disable One Click" option that, when selected, would disable the single input GUI control element.

[0048] The second one click GUI control element 514 may receive the notification signal from the first one click GUI control element 506 and may update in response to the signal. In some implementations, each one click GUI element can send its own signal to its associated subtree. The one click control element that receives the notification can start a new one-click. However, generally a one click GUI element may not initiate a new one-click event in response to a one-click event. Control elements may be responsible for not entering into endless one-click loops.

[0049] FIG. 6 shows a flowchart of an example method for GUI control element processing in accordance with some implementations. Processing begins at 602, where a view tree (e.g., 403, 405, 505) is built (or updated) and where the view tree contains one or more one click GUI control elements. A view tree may be built when a parent window is created and may be updated when an item within a parent window changes (e.g., a view, element, GUI control element is added or removed). Processing continues to 604.

[0050] At 604, the presence of a one click GUI control element is determined within the view tree. For example, the presence of a one click GUI control element could be determined when that element is added to a parent window or is modified within the parent window. Processing continues to 606.
[0051] At 606, a list of selectable items is obtained from a common column. The list of selectable items can form a list from which a user can select or click through using the one click GUI control. The common column refers to a common column (or multiple columns) of a data source (e.g., table) that the one click GUI control element and each of the other views and GUI elements within the parent window which are enabled for one click GUI control have access to and use when updating in response to a notification from the one click GUI control element. For example, the list of selectable items from a common column could include a list of security symbols taken from the security symbol column of a data table. Processing continues to 608.

[0052] At 608, a selected common column value is received from the one click GUI control element. For example, a user may type, click on or otherwise select a security symbol from the list of selectable symbols. Processing continues to 610.

[0053] At 610, the view tree is traversed. The system can traverse the view tree using any suitable technique. Processing continues to 612.

[0054] At 612, as the view tree (or a subtree of the view tree) is traversed, a notification is sent to each node of the view tree (or subtree) corresponding to a view or GUI element in the view tree for the parent window containing the one click GUI control element and for which one click GUI notification listening is enabled. The notification causes those views and GUI elements for which one click GUI control is enabled to update and display potentially different information based on the notification. Processing continues to 614.

[0055] At 614, the one click listener nodes in the view tree (e.g., the nodes associated with views and GUI elements having one click control enabled) obtain updated data based on the selection in the one click GUI element. In some implementations, raw tables can be computed at a remote query processor. The GUI can be a non-primary connection to the remote query processor. When the GUI displays a table, it may retrieve a relevant subset of rows and columns to display. The GUI can also filter the raw table by adding another node to the DAG. Filtering can happen either on the client or server side. In the case of single input GUI control, the value in the single input GUI control element (e.g. Symbol: `AAPL`) can be used to create the viewing filter. Processing continues to 616.

[0056] At 616, the view or GUI element associated with each one click listener node is caused to be updated on a display based on the updated data.

[0057] It will be appreciated that 602-616 can be repeated in whole or in part in order to accomplish a contemplated one click GUI control task.
It will be appreciated that the modules, processes, systems, and sections described above can be implemented in hardware, hardware programmed by software, software instructions stored on a nontransitory computer readable medium or a combination of the above. A system as described above, for example, can include a processor configured to execute a sequence of programmed instructions stored on a nontransitory computer readable medium. For example, the processor can include, but not be limited to, a personal computer or workstation or other such computing system that includes a processor, microprocessor, microcontroller device, or is comprised of control logic including integrated circuits such as, for example, an Application Specific Integrated Circuit (ASIC), a field programmable gate array (FPGA), GPU or the like. The instructions can be compiled from source code instructions provided in accordance with a programming language such as Java, C, C++, C#.net, assembly or the like. The instructions can also comprise code and data objects provided in accordance with, for example, the Visual Basic™ language, a specialized database query language, or another structured or object-oriented programming language. The sequence of programmed instructions, or programmable logic device configuration software, and data associated therewith can be stored in a nontransitory computer-readable medium such as a computer memory or storage device which may be any suitable memory apparatus, such as, but not limited to ROM, PROM, EEPROM, RAM, flash memory, disk drive and the like.

Furthermore, the modules, processes systems, and sections can be implemented as a single processor or as a distributed processor. Further, it should be appreciated that the steps mentioned above may be performed on a single or distributed processor (single and/or multi-core, or cloud computing system). Also, the processes, system components, modules, and sub-modules described in the various figures of and for embodiments above may be distributed across multiple computers or systems or may be co-located in a single processor or system. Example structural embodiment alternatives suitable for implementing the modules, sections, systems, means, or processes described herein are provided below.

The modules, processors or systems described above can be implemented as a programmed general purpose computer, an electronic device programmed with microcode, a hard-wired analog logic circuit, software stored on a computer-readable medium or signal, an optical computing device, a networked system of electronic and/or optical devices, a special purpose computing device, an integrated circuit device, a semiconductor chip, and/or a software module or object stored on a computer-readable medium or signal, for example.
Embodiments of the method and system (or their sub-components or modules), may be implemented on a general-purpose computer, a special-purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit element, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmed logic circuit such as a PLD, PLA, FPGA, PAL, or the like. In general, any processor capable of implementing the functions or steps described herein can be used to implement embodiments of the method, system, or a computer program product (software program stored on a nontransitory computer readable medium).

Furthermore, embodiments of the disclosed method, system, and computer program product (or software instructions stored on a nontransitory computer readable medium) may be readily implemented, fully or partially, in software using, for example, object or object-oriented software development environments that provide portable source code that can be used on a variety of computer platforms. Alternatively, embodiments of the disclosed method, system, and computer program product can be implemented partially or fully in hardware using, for example, standard logic circuits or a VLSI design. Other hardware or software can be used to implement embodiments depending on the speed and/or efficiency requirements of the systems, the particular function, and/or particular software or hardware system, microprocessor, or microcomputer being utilized. Embodiments of the method, system, and computer program product can be implemented in hardware and/or software using any known or later developed systems or structures, devices and/or software by those of ordinary skill in the applicable art from the function description provided herein and with a general basic knowledge of the software engineering and computer networking at s.

Moreover, embodiments of the disclosed method, system, and computer readable media (or computer program product) can be implemented in software executed on a programmed general purpose computer, a special purpose computer, a microprocessor, or the like.

It is, therefore, apparent that there is provided, in accordance with the various embodiments disclosed herein, methods, systems and computer readable media for a GUI control element and associated processing method.

Application No. __________, entitled "DATA PARTITIONING AND ORDERING" (Attorney Docket No. WL 1-10057) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.
[0066] Application No. __________, entitled "COMPUTER DATA SYSTEM DATA SOURCE REFRESHING USING AN UPDATE PROPAGATION GRAPH" (Attorney Docket No. Wl. 4-10058) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0067] Application No. __________, entitled "COMPUTER DATA SYSTEM POSITION-INDEX MAPPING" (Attorney Docket No. Wl. 5-10083) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0068] Application No. __________, entitled "SYSTEM PERFORMANCE LOGGING OF COMPLEX REMOTE QUERY PROCESSOR QUERY OPERATIONS" (Attorney Docket No. W1.6- 10074) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0069] Application No. __________, entitled "DISTRIBUTED AND OPTIMIZED GARBAGE COLLECTION OF REMOTE AND EXPORTED TABLE HANDLE LINKS TO UPDATE PROPAGATION GRAPH NODES" (Attorney Docket No. W 1.8- 10085) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.


[0072] Application No. __________, entitled "DYNAMIC FILTER PROCESSING" (Attorney Docket No. W2.4- 10075) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0074] Application No. __________, entitled "DYNAMIC TABLE INDEX MAPPING" (Attorney Docket No. W2.7-10077) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0075] Application No. __________, entitled "QUERY TASK PROCESSING BASED ON MEMORY ALLOCATION AND PERFORMANCE CRITERIA" (Attorney Docket No. W2.8-10094) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.


[0079] Application No. __________, entitled "DYNAMIC UPDATING OF QUERY RESULT DISPLAYS" (Attorney Docket No. W3.3-10059) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0080] Application No. __________, entitled "DYNAMIC CODE LOADING" (Attorney Docket No. W3.4-10065) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.


[0082] Application No. __________, entitled "COMPUTER DATA DISTRIBUTION ARCHITECTURE" (Attorney Docket No. W3.7-10079) and filed in the United States Patent
and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.


[0085] Application No. __________, entitled "GRAPHICAL USER INTERFACE DISPLAY EFFECTS FOR A COMPUTER DISPLAY SCREEN" (Attorney Docket No. W4.4-10Q90) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.


[0090] While the disclosed subject matter has been described in conjunction with a number of embodiments, it is evident that many alternatives, modifications and variations would be,
or are, apparent to those of ordinary skill in the applicable arts. Accordingly, Applicants intend to embrace all such alternatives, modifications, equivalents and variations that are within the spirit and scope of the disclosed subject matter.
CLAIMS

What is claimed is:

1. A method of controlling a graphical user interface (GUI) having a plurality of elements based on input from a single GUI control element, the method comprising:
   providing a view tree containing nodes corresponding to graphical user interface (GUI) elements within a parent window;
   determining presence of a first GUI control element in the view tree of the parent window;
   receiving an event from the first GUI control element, the event corresponding to a single input event to the GUI control element;
   in response to the received event, traversing the view tree; and
   at each node of the view tree reached during the traversing:
      sending a notification signal of the event,
      when a node has listening for events from the first GUI control element enabled, updating the GUI element associated with the node based on the GUI control element notification signal; and
      causing the updated GUI element to be displayed.

2. The method of claim 1, wherein the GUI element can include one of a view and a GUI element.

3. The method of claim 1, further comprising disabling listening for a notification signal from the first GUI control element at a node when an indication is received to disable GUI control element listening for that node.

4. The method of claim 1, further comprising updating the view tree when a GUI element is added to, removed from, or moved within the parent window or removed from the parent window.

5. The method of claim 1, further comprising providing a second GUI control element within the parent window, the second GUI control element configured to listen for notifications from the first GUI control element.
6. The method of claim 1, wherein the view tree includes a partial view tree.

7. A system for controlling a graphical user interface (GUI) having a plurality of elements based on input from a single GUI control element, the system comprising:

one or more hardware processors coupled to a nontransitory computer readable medium having stored thereon software instructions that, when executed by the one or more processors, cause the one or more processors to perform operations including:

providing a view tree containing nodes corresponding to graphical user interface (GUI) elements within a parent window;

determining presence of a first GUI control element in the view tree of the parent window;

receiving an event from the first GUI control element, the event corresponding to a single input event to the GUI control element;

in response to the received event, traversing the view tree; and

at each node of the view tree reached during the traversing:

sending a notification signal of the event;

when a node has listening for events from the first GUI control element enabled, updating the GUI element associated with the node based on the GUI control element notification signal; and

causing the updated GUI element to be displayed.

8. The system of claim 7, wherein the GUI element can include one of a view and a GUI element.

9. The system of claim 7, wherein the operations further include disabling listening for a notification signal from the first GUI control element at a node when an indication is received to disable GUI control element listening for that node.

10. The system of claim 7, wherein the operations further include updating the view tree when a GUI element is added to, removed from, or moved within the parent window or removed from the parent window.

11. The system of claim 7, wherein the operations further include providing a second GUI
control element within the parent window, the second GUI control element configured to listen for notifications from the first GUI control element.

12. The system of claim 7, wherein the view tree includes a subtree.

13. A nontransitory computer readable medium having stored thereon software instructions that, when executed by one or more processors, cause the one or more processors to perform operations including:
   providing a view tree containing nodes corresponding to graphical user interface (GUI) elements within a parent window;
   determining presence of a first GUI control element in the view tree of the parent window;
   receiving an event from the first GUI control element, the event corresponding to a single input event to the GUI control element;
   in response to the received event, traversing the view tree; and
   at each node of the view tree reached during the traversing:
       sending a notification signal of the event;
   when a node has listening for events from the first GUI control element enabled, updating the GUI element associated with the node based on the GUI control element notification signal; and
   causing the updated GUI element to be displayed.

14. The nontransitory computer readable medium of claim 13, wherein the GUI element can include one of a view and a GUI element.

15. The nontransitory computer readable medium of claim 13, wherein the operations further include disabling listening for a notification signal from the first GUI control element at a node when an indication is received to disable GUI control element listening for that node.

16. The nontransitory computer readable medium of claim 13, wherein the operations further include updating the view tree when a GUI element is added to, removed from, or moved within the parent window or removed from the parent window.
17. The nontransitory computer readable medium of claim 13, wherein the operations further include providing a second GUI control element within the parent window, the second GUI control element configured to listen for notifications from the first GUI control element.

18. The nontransitory computer readable medium of claim 13, wherein the view tree includes a subtree.
FIG. 2
FIG. 3
FIG. 4
Build/Update View Tree Containing One or More Single Input GUI Control Elements

Determine Single Input GUI Control Element Presence Within View Tree

Obtain List of Selectable Items from Common Column

Receive Selected Common Column Value at Single Input GUI Control Element

Traverse View Tree (or subtree)

Send Notifications to Each Node in View Tree

Single Input Listener Nodes Update Data Associated with View Based on Selection in Single Input GUI Control Element

Single Input Listener Nodes Update Corresponding View Based on Updated Data

FIG. 6
INTERNATIONAL SEARCH REPORT

INTERNATIONAL SEARCH REPORT

PCT/US 2016/032602

A. CLASSIFICATION OF SUBJECT MATTER

G06F3/048 (2006.01)
G09G5/14 (2006.01)

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)

G06F3/00-3/048, 3/14, 9/00, 9/44, 15/00, 15/16, 17/00, 17/30, G09G5/00, 5/14

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

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

DWPI, ESP®CENET, K-PION, PatSearch, RUPTO, USPTO

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>X</td>
<td>US 2005/0102636 A1 (MICROSOFT CORPORATION), 12.05.2005, abstract, [0046], [0051H0064], [0070], [0082]-[0084], [0096], claim 1, fig. 3, 11A-1 IC</td>
<td>1-18</td>
</tr>
<tr>
<td>A</td>
<td>CA 2309462 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION), 03.12.2000</td>
<td>1-18</td>
</tr>
</tbody>
</table>

* Special categories of cited documents:

- **A** document defining the general state of the art which is not considered to be of particular relevance
- **E** earlier document published on or after the international filing date
- **L** document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- **O** document referring to an oral disclosure, use, exhibition or other means
- **P** document published prior to the international filing date but later than the priority date claimed

- **T** later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- **X** document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- **Y** document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
- **&** document member of the same patent family

Date of the actual completion of the international search: 12 August 2016 (12.08.2016)

Date of mailing of the international search report: 01 September 2016 (01.09.2016)

Name and mailing address of the ISA/RU:
Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993
Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer
R. Lopatkina

Telephone No. (8-499) 240-25-91

Form PCT/ISA/210 (second sheet) (January 2015)