159159

APPARATUS AND METHOD FOR IDENTIFYING A WINDOW

Technical Field

The invention relates to an apparatus, a method and a computer program product having instructions for performing a method for identifying a lowest ranked window of display windows in a distributed control system.

Background

Distributed Control Systems (DCSs) such as an integrated production control system typically comprise a human interface station or a console for plant operation and monitoring by a user. Typical DCSs are dedicated systems used to control manufacturing processes that are continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking. DCSs are typically connected to sensors and actuators and used to control and/or monitor a process flow.

The station provides a human machine interface that displays a control application executed by a computer. The control application may be displayed on a screen as a number of different functions or applications in a multiple-window mode as shown in Fig. 1, where multiple display windows 102, 104, 106, 108 (each displaying the contents of a function or application) are opened, and a minimized window is represented as a taskbar icon 110 in an exemplary screen shot 100. Alternatively, the various functions or applications that have been called up may be displayed one at a time on a panel, with tabs allowing the selection of a particular function or application for current display on the panel.

Hereon, the word "window" is taken to include an opened window in a multiple-window mode display, a minimized window represented by a taskbar icon, and also a tab for selecting a function or application in a panel mode display.

To maximize performance efficiency, the control application/distributed control system normally has a window manager configured to manage the display of windows on a

G00002

screen of the human/machine interface. One task this windows manager performs is to minimize the time taken between the user calling up a function and loading content of that function into a display window. This is sometimes achieved by limiting the maximum number of display windows that can be on the screen at any time. Limiting the maximum number prevents excessive memory usage and avoids a situation of a function not being able to be called up for lack of memory in the computer.

During operation, the user may wish to call up several functions or applications at once. Because there is only a maximum number of display windows that can be on the screen at any one time, the window manager will overwrite one of the existing windows (or the content of the window) with new content if a new function is called after the maximum number of windows are already on the screen.

Normally, the window that will be overwritten is identified by the window manager as a lowest ranked window, e.g. an earliest called window among those already on the screen. The user may be able to tell which window this will be by seeing how the windows are overlapped. For example, in the screen shot of Fig. 1, the lowest ranked window appears to be window 108. Some applications offer user functionality so that, if the user wants to keep the contents of window 108 on the screen, the user uses a pin function in the distributed control system (or window manager thereof) to pin window 108 and thereby secure window 108 from being overwritten by new content. If a new function is then called up, the window manager does not overwrite the content of the pinned window, but selects the next lowest ranked window for display of the content of the new function.

However, depending on how a user moves and accesses the display windows on the screen, it may be difficult or inconvenient for the user to tell which window will be erased next when there is already the maximum number of display windows on the screen and a new function is called. For example, minimized window 110 may in fact have been called before window 108. A user who had wanted to keep the contents of window 110 on the screen may find those contents overwritten when a new function is called if window 110 had not been pinned from being erased.

Summary

The invention is defined in the independent claims. Some optional features of the invention are defined in the dependent claims.

The disclosed technique alerts a user to a lowest ranked window among a maximum number of windows on a screen, by changing a display characteristic of that window. Thus, the lowest ranked window is more easily distinguished by the user from the other windows on the screen.

Alerting the user to the lowest ranked window provides enhanced functionality. One advantage of this enhanced functionality is that the user is alerted to a situation where a lowest ranked window (or the content thereof) will be overwritten when a new function is called. If the lowest ranked window is a window the user does not wish to be overwritten the user may take preventive action to ensure the window/window content is not overwritten. Non-exhaustive examples of this are:

- the user can close other non-prioritised windows thus meaning a maximum number of display windows are no longer open. Thus, when a new function or window is called, no window/window content is to be overwritten for that new content;
- the user can manipulate the windows so that another window becomes the lowest ranked window. One example of doing this would be for the user to select (e.g. by clicking a mouse pointer on) that window. This window would then no longer be the lowest ranked window having been the latest one to be active.
- the user can secure the lowest ranked window from it or its content being overwritten. One manner of doing this would be to use a 'pin' function – as discussed above – to pin the window and secure. it from being inadvertently overwritten by new content if the user were to call up another function when there is already the maximum number of windows on the screen.

Brief Description of the Drawings

The invention will now be described, by way of example only, and with reference to the accompanying figures in which:

Fig. 1 is an exemplary screen shot from a display screen of a human machine interface in a multiple-window mode;

Fig. 2 is an exemplary process flow diagram providing an overview of a technique for identifying a lowest ranked window;

Fig. 3 is another exemplary process flow diagram providing an overview of a technique for identifying a lowest ranked window;

Fig. 4 is a block diagram illustrating an architecture for an apparatus for identifying a lowest ranked window;

Fig. 5 is an exemplary process flow diagram providing an overview of a technique for identifying a lowest ranked window; and

Fig. 6 is an exemplary screen shot from a display screen of a human machine interface in a tabbed display mode.

Detailed Description of the Preferred Embodiments

Exemplary embodiments of a method as outlined in Fig. 2, Fig. 3 and Fig. 5, and an exemplary apparatus shown in Fig. 4, will now be described.

Firstly, it is convenient to discuss the apparatus of Fig. 4. Apparatus 400 principally comprises a window manager 406. The window manager 406 of apparatus 400 may interact with and be supported by ancillary architecture modules memory 408, graphical user interface (GUI) 410 and user input device (UID) 412. In one implementation, GUI has a display generator (not shown), or interacts with a display generator, to display windows on a display (not shown) to a user.

Optionally, apparatus 400 comprises a control application module 402 providing a pin function as will be discussed below, either as part of module 402 or as a pin function sub-module 404.

The control application 402 provides a number of functions or applications that can be called up by a user. Each function or application may be opened in a window or as a tabbed panel. Pin function module 404 of control module 402 provides the pin function for securing a window from being overwritten.

As will be described in detail with reference to Figures 2, 3 and 5, window manager 406 is configured to determine that a maximum number of display windows has been opened in the control application 402. The maximum number may be preset into the control application 402, or may be determined by a user and keyed in through the UID 412 which may, for instance, be a keyboard, mouse, or an alphanumeric pad.

The window manager 406 determines when the number of display windows opened is not at a maximum. In such a case, when a user calls up another function or application the window manager opens a new display window and loads new content into the new display window. In this way, new windows each containing a function or application can be opened until the maximum number of windows are on the screen.

The window manager 406 is further configured to identify the lowest ranked window of the display windows if it determines that the maximum number of windows has been opened for display. The window manager 406 is also configured to change a display characteristic of the lowest ranked window. In implementations when the pin function module 404 is provided, the window manager is configured to determine whether the lowest ranked window is pinned and, if so, identify the next lowest ranked window, and so forth, until a display characteristic of the lowest ranked unpinned window has been changed.

The window manager 406 is also configured to restore normal display characteristics to the lowest ranked unpinned window upon closing one of the maximum number of display windows.

Referring to Figure 5, a process flow of a first technique for identifying a lowest ranked window of display windows of a distributed control systems is discussed. Process 500 starts at step 502. At step 504, window manager 406 makes a determination of whether the maximum number of windows has been opened for display. If window manager 406 determines that a maximum number of display windows has been opened, then window manager 406 identifies, at step 506, a lowest ranked window of the display windows. The options for making this determination are discussed below. Once window manager

406 identifies the lowest ranked window, a display characteristic of the lowest ranked window is changed at step 508. The process ends at step 510.

Referring to Fig. 2, a process flow of a second technique for identifying a lowest ranked window of display windows in a distributed control system is illustrated. Process 200 starts at step 202 where no windows are currently open in the control application on the screen. At step 204, window manager 406 waits for a function to be called to open a window for that function. If a window is to be opened, window manager 406 determines at step 206 whether a maximum number of windows is already open for display. If the maximum number of windows has not yet been opened, at step 208, window manager 406 opens a new display window and loads new content into that new window. The process loops around steps 204, 206, and 208 with the user being able to open windows on the screen until, at step 206, window manager 406 determines that a maximum number of windows has been opened. The window manager then identifies the lowest ranked window of the display windows. In one implementation, window manager 406 sets that window which is the earliest (i.e. first) called window among the display windows as the lowest ranked window at step 210.

At step 212, window manager 406 determines whether the lowest ranked window is secured from being overwritten by, for example, use of pin function module 404. If the lowest ranked window has been pinned, window manager 406 knows a user has secured this window (or its contents) from being overwritten and identifies the next lowest ranked window (e.g. the second-called window) at step 214. The process loops around steps 212 and 214 until window manager 404 identifies a lowest-ranked window which is not secured from being overwritten, whereupon at step 216 window manager 406 changes a display characteristic of that window.

Changing a display characteristic of a window may include rendering that window semitransparent, rendering the tab for the window semi-transparent as illustrated by tab 64 in the series of window tabs 61, 62, 63, 64 of Fig. 6, changing the colour of the title bar or border of that window, having that window flash on and off intermittently, or any other appropriate change for the purpose of alerting the user to the fact that window (or

its contents) will be overwritten at the next instance a function is called which requires a new window to be called or new window content to be displayed. .

At step 218, upon being alerted to a window because a display characteristic of that window has been changed, that window may become pinned. This could either be performed manually by a user, or an action which is executed automatically by, say, window manager 406. If the lowest ranked window is pinned, the window manager will restore normal display characteristics of that window at step 220, and reverts to step 214 to identify the next lowest-ranked, un-pinned window.

The process loops around steps 212, 216, 218, 220 and 214 until the window is not pinned at step 218. A determination is then made at step 222 whether or not to call up a new function, that is, to open a window or not. If no window is to be opened at step 222, the screen will continue to show the maximum number of windows with one of the windows having a changed display characteristic, indicating that this is the window that will be overwritten if a new function is called.

If a window is to be opened, the window manager will load content of the newly called function into the window with the changed display characteristic at step 224, and also restore normal display characteristics to that window. From step 224, the window manager then determines at step 226 whether the maximum number of windows is still on the screen. If the user closes a window, the process goes back to step 204 to wait for a determination to open a window or not. If the maximum number of windows remains on the screen, the process loops back to step 210 in order to find a lowest-ranked un-pinned window and change a display characteristic of that window.

In this way, as long as there is a maximum number of windows open on the screen, one of the windows will have a changed display characteristic indicating that it is the window that will be overwritten if a new function is called. The pin function allows an overriding of the window manager's choice of window to be overwritten. This compels the window manager to find another window to be overwritten.

Fig. 3 shows an alternative exemplary overview of the disclosed method for identifying a lowest ranked window. The difference between process 300 and the process 200 of Fig. 2 is that the process 300 starts at step 302 with a maximum number of display windows already on the screen and with one of the windows already having a display characteristic changed by the window manager.

At step 304, a determination is made whether or not to call up a function, that is, to open a window or not. If no window is to be opened, a determination is made at step 322 whether a window has been closed. If no window is closed, the process goes back to the start at step 302 and there remains the maximum number of windows on the screen with one of the windows having a changed display characteristic.

If a window has been closed, a determination is made at step 324 whether one of the remaining windows has a changed display characteristic. If there is no such window, this means that the window that was closed was the window with the changed display characteristic and the process loops back to the start at 302. If one of the remaining windows has a changed display characteristic, the window manager will restore normal display characteristics to that window at step 326 and go back to the start at 302. This is because a window has been determined to be closed at step 322, so the number of windows on the screen is no longer the maximum and therefore no window needs to be marked out for potential overwriting.

If at step 304 it is found that a window is to be opened, the window manager determines at step 306 whether the maximum number of windows is still on the screen. If there is a maximum number of windows on the screen (also meaning that one of the windows has a changed display characteristic), the window manager will load the new content into the display window with the changed display characteristic at step 308.

If there is no longer the maximum number of windows opened (e.g., the user closed a window at step 322), then at step 310, the window manager opens a new display window and loads content of the newly called function into that new window. The

window manager then determines in step 312 whether the maximum number of windows has been reached.

If the maximum number is not reached, the process goes back to the start 302 and loops around steps 304, 306, 310, 312 and 302 until it is determined at step 312 that there is now the maximum number of windows on the screen. Whereupon the window manager at step 314 identifies the lowest ranked window among all the windows on the screen. This is the earliest-called window among those already on the screen.

At step 316, the window manager determines whether the lowest ranked window has been pinned and is thus secured from being overwritten. If that window has been pinned, the window manager will identify the next lowest ranked window at step 320. The process loops around steps 316 and 320 until the window manager finds a lowest-ranked un-pinned window, whereupon at step 318 the window manager changes a display characteristic of that window. There is again the maximum number of windows on the screen with one of the windows having a changed display characteristic. The process reverts to the starting step 302.

It will be appreciated that the apparatus disclosed herein may be, for example, one or more computer apparatus. The various techniques disclosed may be implemented in hardware, software modules loaded on the apparatus or a combination thereof.

It will be further appreciated that the invention has been described by way of example only and that variations in detail may be made without departure from the spirit and/or scope of the appended claims.

Claims

1. An apparatus for identifying a lowest ranked window of display windows of a distributed control system, the apparatus comprising a window manager configured to:

determine a maximum number of display windows has been opened;

identify a lowest ranked window of the display windows; and

change a display characteristic of the lowest ranked window.

2. The apparatus of claim 1, wherein the window manager is configured to load new content into the lowest ranked display window.

3. The apparatus of any one of the preceding claims, wherein the window manager is configured to indentify the lowest ranked window as an earliest opened window.

4. The apparatus of any one of the preceding claims, wherein the window manager is configured to indentify the lowest ranked window has been secured from being overwritten or erased, and identify a next lowest ranked window as a new lowest ranked window.

5. The apparatus of any one of the preceding claims, wherein changing the display characteristic includes rendering the lowest ranked window semitransparent.

6. The apparatus of any one of the preceding claims, wherein the window manager is configured to determine when the number of display windows opened is no longer the maximum and to open a new display window and load new content into the new display window if the number of display windows opened is no longer the maximum.

7. The apparatus of any one of the preceding claims, wherein the window manager is configured to restore normal display characteristics to the lowest ranked window upon closing one of the maximum number of display windows.

8. A method for identifying a lowest ranked window of display windows of a distributed control system, the method comprising:

determining a maximum number of display windows has been opened;

identifying a lowest ranked window of the display windows; and

changing a display characteristic of the lowest ranked window.

9. A method for identifying a lowest ranked window of display windows of a distributed control system using the apparatus of any one of claims 1 to 7.

10. A computer program product having computer program code stored thereon or embedded therein comprising executable instructions for implementing the method of any of claims 8 to 9.