EXHIBIT 1.09

As shown in FIG. 1, a pen-based computer system 10 in accordance with the present invention includes a central processing unit (CPU) 12, read only memory (ROM) 14, random access memory (RAM) 16, input/output (I/O) circuitry 18, and a display assembly 20. The pen-based computer system 10 may also optionally include a mass storage unit 22 such as a disk drive unit or nonvolatile memory such as flash memory, a keypad 24, and a clock 26.

The CPU 12 is preferably a commercially available, single chip microprocessor. While CPU 12 can be a complex instruction set computer (CISC) chip, it is preferable that CPU 12 be one of the commercially available, reduced instruction set computer (RISC) chips which are known to be of generally higher performance than CISC chips. CPU 12 is coupled to ROM 14 by a unidirectional data bus 28. ROM 14 contains the basic operating system for the pen-based computer system 10. CPU 12 is connected to RAM 16 by a bi-directional data bus 30 to permit the use of RAM 16 as scratch pad memory. ROM 14 and RAM 16 are also coupled to CPU 12 by appropriate control and address busses, as is well known to those skilled in the art. CPU 12 is also coupled to the I/O circuitry 18 by bi-directional data bus 32 to permit data transfers with peripheral devices.

I/O circuitry 18 typically includes a number of latches, registers and direct memory access (DMA) controllers. The purpose of I/O circuitry 18 is to provide an interface between CPU 12 and such peripheral devices as display assembly 20, mass storage 22, and the keypad 24.

Clock 26 provides a series of clock pulses and is typically coupled to an interrupt port of CPU 12 by a data line 34. The clock pulses are used to time various functions and events relating to the computer system 10. The clock 26 can be eliminated and the clock function replace by a software clock running on CPU 12, but this tends to be a wasteful use of CPU processing power. In the present invention, clock 26 provides clock pulses at 60 hertz (Hz).

Display assembly 20 of pen-based computer system 10 is both an input and an output device. Accordingly, it is coupled to I/O circuitry 18 by a bi-directional data bus 36. When operating as an output device, the display assembly 20 receives data from I/O circuitry 18 via bus 36 and displays that data on a suitable screen. The screen for display assembly 20 is preferably a liquid crystal display (LCD) of the type commercially available from a variety of manufacturers. The input device of display assembly 20 is preferably a thin, clear membrane which covers the LCD display and which is sensitive to the position of a stylus 38 on its surface. These position sensitive membranes are also readily available on the commercial market. Combination display assemblies such as display assembly 20 which include both the LCD and the input membrane are commercially available from such vendors as Scriptel Corporation of Columbus, Ohio.

The keypad 24 can comprise an array of switches. In the present embodiment, the keypad 24 comprises mechanical buttons which overlie the bottom edge of the membrane which covers the LCD display. When the buttons are depressed, the membrane senses the pressure and communicates that fact to the CPU 12 via I/O 18.

Other types of pointing devices can also be used in conjunction with the present invention. While the method of the present invention is described in the context of a pen-based system, other pointing devices such as a computer mouse, a track ball, or a tablet can be used to manipulate a pointer on a screen of a general purpose computer. Therefore, as used herein, the terms "pointer", "pointing device", "pointing means", and the like will refer to any mechanism or device for pointing to a particular location on a screen of a computer display.

Some type of mass storage 22 is generally considered desirable. However, the mass storage 22 can be eliminated by providing a sufficient amount of RAM 16 to store user application programs and data. In that case, the RAM 16 could be provided with a backup battery to prevent the loss of data even when the pen-based computer system 10 is turned off. However, it is generally desirable to have some type of long term storage 22 such as a commercially available miniature hard disk drive, nonvolatile memory such as flash memory, battery backed RAM, PC-data cards, or the like.

In operation, information is input into the pen-based computer system 10 by "writing" on the screen of display assembly 20 with the stylus 38. Information concerning the location of the stylus 38 on the screen of the display assembly 20 is input into the CPU 12 via I/O circuitry 18. Typically, this information comprises the Cartesian (i.e. x & y) coordinates of a pixel of the screen of display assembly 20 over which the tip of the stylus is positioned. Commercially available combination display assemblies such as the aforementioned assemblies available from Scriptel Corporation include appropriate circuitry to provide the stylus location information as digitally encoded data to the I/O circuitry of the present invention. The CPU 12 then processes the data under control of an operating system and possibly an application program stored in ROM 14 and/or RAM 16. The CPU 12 next produces data which is output to the display assembly 20 to produce appropriate images on its screen.

In FIG. 2, the pen-based computer system 10 is shown housed within a generally rectangular enclosure 40. The CPU 12, ROM 14, RAM 16, I/O circuitry 18, mass storage 22, and clock 26 are preferably fully enclosed within the enclosure 40. The display assembly 20 is mostly enclosed within the enclosure 40, but a viewing screen 42 of the display assembly is exposed to the user. As used herein, the term "screen" will refer to the portion of the display assembly 20 which can display an image that can be viewed by a user. Also accessible to the user is the keypad 24.

Upon power-up, pen based computer system 10 displays on screen 42 an initial note area N including a header bar B and a number of guidelines 44. The header bar B preferably includes the date of creation 46 of the note N, a note number 48, and a "toolbox" button 50 represented by a toolbox icon. The optional guidelines 44 aid a user in entering text, graphics, and data into the pen-based computer system 10.

In this preferred embodiment, the keypad 24 is not a part of the viewing screen 42 but rather, is a permanent array of input buttons coupled to the CPU 12 by I/O circuitry 18. Alternatively, the keypad 24 could comprise "soft buttons" generated at a convenient location on the screen 42, in which case a "button" would be activated by touching the stylus to the screen over the image of the button. The keypad 24 preferably includes a number of dedicated function buttons 52 and a pair of scroll buttons 54A and 54B. The operation of the scroll buttons 54A and 54B, and other aspects of computer system 10 are discussed in greater detail in co-pending U.S. patent application Ser. No. 07/868,013, filed Apr. 13, 1992, now U.S. Pat. No. 5,398,310, on behalf of Tchao et al. and entitled "Method for Manipulating Notes on a Computer Display". That application is assigned to the assignee of the present application and its disclosure is hereby incorporated by reference in its entirety. In this

embodiment, the toolbox button **50** is represented as a "soft button" in the header bar B. However, in alternative embodiments, a permanent, hardwired keypad button could be used in its place.

The screen illustrated in FIG. **2** is referred to as the "notepad", and is an application program running under the operating system of the pen based computer system **10**. In this preferred embodiment, the notepad is a special or "base" application which is always available beneath higher level applications. The notepad application, like other applications, run within a window, which in this instance comprises the entire viewing screen **42**. Therefore, as used herein, a "window" is the entire screen or any portion of an entire screen which is dedicated to a particular application program.

A status bar **56** is provided at the bottom of the notepad application. The status bar **56** is provided with a number of active areas including a real time clock **58**, a view button **60**, a font button **62**, a formulas button **64**, a text button **66**, a graphics button **68**, and a nib button **70**.

The real time clock **58** is an example of a "global" active area which derives information or controls of function not necessarily associated with the application window to which it is attached. The buttons **60** and **62** are examples of active areas which provide indirect control over the notepad function. For example, pressing the font button **62** will pop up a window providing a selection of fonts which can be used when writing within the notepad window. Buttons **64–70** are examples of active areas which provide direct control over the notepad application. For example, the button **70** controls the nib size of the "ink" produced by the stylus within the application window area. Buttons **64–68** aid in recognizing writings made in the window area.

FIG. **3** illustrates a window produced by a "cardfile" application program. U.S. patent application Ser. No. 07/955,839 filed Oct. 2, 1912, now U.S. Pat. No. 5,446,882, on behalf of Capps et al., entitled "Computerized Database With Card & List Interface" and assigned to the assignee of the present invention describes the operation of such a cardfile application, and is incorporated herein in its entirety by reference.

In FIG. **3**, the screen **42** is provided with a window **72** associated with the card file application. In this instance, the window assumes the size and shape of a business card and displays the name, company and telephone number of a Mr. Gregg Foster. It should be noted that the window **72** is considerably smaller than the screen **42** in this example, leaving the possibility of additional windows being opened on the screen **42** in an overlapping or non-overlapping fashion.

Status bar **74** is attached to the bottom of window **72** by the process of the present invention. The status bar **74** includes a number of active areas including a close box **76**, a real time clock **78**, a filter button **80**, a "more" button **82**, a "new" button **84**, and a routing slip **86**. The close box **76** is an active area which permits the card file application to be closed, i.e. is either "made invisible" or is completely deactivated. The real time clock **78** is, once again, an active area carrying global information which is not necessarily associated with that particular application. The filter button **80** does not act directly upon the card file application but, rather, pops up a menu of filter parameters (e.g., business, personal, etc.) and permits a user to choose one of the filters to act upon the application program. The more button **82** is an active area which controls the card file application program to permit more or less information to be displayed on the screen **42**. The new button **84** is an active area which permits a new "business card" to be entered into the card file application.

The function of the routing slip button **86** will be discussed in greater detail with reference to FIG. **4**. When a user presses the routing slip button **86**, a window **88** pops up from the status bar **74** to provide a list of options concerning the desired disposition of the information displayed within window **72**. For example, the information in window **72** can be faxed by pressing on "fax" within window **88**, can be deleted by pressing "trash" within window **88**, etc.

FIG. **5** illustrates the result if "fax" is chosen from window **88** of FIG. **4**. The selection of "fax" within the window **88** starts an application program which is provided with a floating window **90** which partially overlies the window **72** of the card file application. The window **90** includes a number a fields to be filled in by the user including a who, what, where, and a number field. The window **90** also permits the user to indicate the format of the facsimile, which in this case, is "card".

Attached to the bottom of the fax application window **90** is a status bar **92**. The status bar has four active areas including a closing box **94**, an information area **96**, a preview button **98**, and a "do it" button **100**. The close box **94** operates as in previous examples, i.e., it closes the window **90** for the fax application program. The information area **96** displays information concerning the fax application; in this case, it displays the title of the application program. Preview button at **98** permits a preview of the image that is to be faxed, and the do it button **100** starts the faxing process. If a user activates the closing box before activating the do it button, the fax will not be sent.

FIG. **6** illustrates a flow diagram of a process **102** for providing a status bar for application windows. The process begins at **104** and starts a new application program in a step **106**. Next, in a step **108**, a status bar is coupled to the application program window. After the status bar is coupled to the application program window, the application program window with status bar is displayed on a computer screen in a step **110**. Next, in a step **112**, the process determines whether there is any feedback from the application program to the status bar. If there is, the appropriate area oil the status bar is updated in a step **114**. If there is no application feedback, is determined whether there is any status bar action in a step **116**. If there is, then the status bar action is processed in a step **118**. If there is no status bar action detected in step **116**, then it is determined whether if there was an button action in step **120**. If so, then the button action is processed in the step **122**. If there is no application feedback (step **112**), status bar action (step **116**), or button action (step **120**) then process control is returned to step **110** or the application program continues to be displayed on the screen. After the completion of steps **114**, **118**, and **122**, process control is also returned to step **110**.

It should be noted that the flow diagram of FIG. **6** is a conceptual representation of the functioning of the process of the present invention, but that the process can be implemented in a variety of manners. For example, the process steps **112**, **116**, and **120** are preferably accomplished in parallel within the context of a "view system". In such a view system, various "views" or "objects" are stacked on top of each other, like pages of paper on a desk top. These views include a root view (such as the notepad) and virtually any number of views (within the limitations of the system) stacked on top of the root view.

The view system is a software routine which returns two pieces of information when the screen "tapped" by a stylus.

A first piece of information returned which view or "object" was tapped. The second piece of information returned is the position of the tap on the tapped view. This location information is often returned in the form of Cartesian (x-y) coordinates. The view system therefore handles much of the routine input work for the computer system. Taps by stylus on non-active areas of the screen can be ignored by the view system. Likewise, inappropriate inputs on active areas of the screen can be ignored or can generate error conditions which may or may not be acted upon by the system.

The flow diagram of FIG. **7** illustrates the "Couple Status Bar" step **108** of FIG. **6** in greater detail. The process starts at **124**, and in a step **126** the application obtains a status bar template. The application determines whether there are any areas to fill in the template in a step **128** and, if there are, it fills the areas in a step **130**. If there are not any areas to fill in step **128** or after the completion of step **130**, the step **108** is completed as indicated at **132**.

FIGS. **8A–8D** illustrate a collection of templates which can be used by different application programs. These templates are preferably provided in the form of "objects". As is well known to software programmers, an "object" is a logical software unit comprising data and processes which give it capabilities and attributes. For example, an object can be queried as to its type and can return such data as the number of words that it contains. Objects can contain other objects of the same or of a different type. Objects can also be used to project images on a screen according to their object type. There are many well know texts which describe object oriented programming. See, for example, Object Oriented Programming for the Macintosh, by Kurt J. Schmucher, Hayden Book Company, 1986.

In FIG. **8A**, a generic or base template **134** is simply defined by a height H and a width W. An application program can provide any desired active areas it wishes anywhere in contact with the template **34**.

In FIG. **8B**, a template **136** which is used by the notepad application has one predefined active area **138** corresponding to the real time clock. Therefore, every time that the notepad application is opened, the status bar associated with the notepad application window will necessarily include real time clock **138**. Other active areas can be specified by the notepad application.

In FIG. **8C**, another template **140** is provided with two predefined active areas **142** and **144** corresponding to a close box and a real time clock, respectively. This template **140** is associated with such applications as the aforementioned card file application.

In FIG. **8D**, a template **146** is provided with two predefined active areas **148** and **150**. This template **146** is associated with the routing slip and active area **148** is the close box and area **150** is a mandatory information area. The remaining portions of the template **146** can be filled in as desired by the routing slip application.

Preferably, there are a number of different, specialized status bar templates available to various applications. Alternatively, a single, generic status bar template can be provided, at the cost of increased work in each application program to customize the status bar for its own use. Therefore, the step **126** of FIG. **7** can comprise either choosing one of a plurality of templates or choosing the only template that is provided.

FIG. **9** is a flow diagram illustrating the "Fill Areas" step **130** of FIG. **7** in greater detail. The step **130** begins at **152** and asks in a step **154** whether all the areas have been filled. In the first time through the loop of step **130**, the answer for this would, of course, be "no". In this instance a new area would be created in step **156** and the new area would be installed in a step **158**. Process control is then returned to step **154**. When all areas have been filled, the step **130** is completed as indicated at **160**.

In FIG. **10**, the step **156** "Create a New Area" of FIG. **9** is illustrated in greater detail. The step **156** begins at **162** and a step **164** determines whether a button is to be created. If so, a button template is obtained in a step **166**, and is customized in a step **168**. The process would then be completed as indicated at **170**. If step **164** determines that a button is not to be created, a step **172** determines whether a label is to be created. If so, a text template is obtained in a step **174**, the text template is customized in a step **176**, and the step **156** is completed as indicated at **170**. If neither a button or label is to be created, a step **178** determines whether there are any other types of areas to be created. If so, a template is retrieved in a step **180** and customized in a step **182**. If step **178** determines that there are not any other areas to create or after the completion of step **182**, the step **156** is completed as indicated at **170**.

As seen in FIG. **11**, after step **130** of FIG. **9** is completed, a status bar **184** is provided including a template **186** and a number of active areas **188**, **190**, **192**, **194**, **196**, and **198**. The status bar **184** can be handled as a single object comprising a template object **186** and a number of button, label, or other area objects **188–198**. This status bar object **184** is then ready to be attached or coupled to an application program window as indicated in step **108** of FIG. **6**.

Since the status bar **184** has been customized for and attached to a particular application window, it is completely unambiguous as to which window that status bar controls. Nonetheless, the status bars provide a common interface in that they are all similar at least at the generic template level. A user will always know that an application window will have a status bar attached to it and that certain similar application programs will have similar active areas. For example, all status bars will include a "close box" except for the status bar for the notepad.

The use of object oriented programming and the aforementioned view system simplifies the implementation of the process of the present invention. In FIG. **12A**, a conceptual representation of various objects is shown. The screen **42** forms a base or root layer, and a window **200** associated with an application program forms a second layer above the base or root layer. The template **186** is positioned over the window object **200**, and the various areas **188–198** are positioned over the template **186**.

In FIG. **12B**, a side elevation taken along line **12*b*–12*b*** of FIG. **12A** again illustrates the conceptual layering of various objects. The aforementioned viewing system automatically handles "taps" of the stylus **38** on the screen **42** by returning information concerning which object has been tapped and where on the object that the tap occurred. For example, a tap A on the screen **42** would create an action for whatever application was displayed on that area of the screen. For example, the tap might activate the notepad on the screen **42**. A tap B on the window **200** could potentially cause an interaction with the application program being displayed within that window **200**. A tap C on the template **164** could be part of a gesture formed on that template or it could be of part of a drag action. A tap D on close button **188** would cause the application window **200** to close. It is therefore clear that the object oriented programming and view system software makes the implementation of the process **102** of FIG. **6** a relatively less cumbersome process than traditional programming techniques.

In FIG. **13**, the "Do Status Bar Action" step **118** of **6** is illustrated in greater detail. The process **118** begins at **202** and step **204** determines whether a drag action is being performed. A drag action could be indicated by placing the tip of the stylus **38** on a portion of the template and then moving the tip of the stylus across the screen **42**. This would cause both the status bar and the attached window to move in a step **206**. If the drag action is not being indicated, it is determined in a step **208** whether a gesture action is being indicated. An example of a gesture is a "scrub" gesture which could close the window. If a gesture is being indicated, then step **210** performs the gesture. Other actions might be detected in a step **212** in which case other actions could be taken in a step **214**. After the completion of steps **206, 210, 214**, or if recognizable meaning is associated with status bar action **118**, the process is completed as indicated at **212**.

While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. For example, the status bar could take other forms than a bar, e.g. it could take the form of a "status slip" of any convenient shape. Also, the status bar could be attached to the top or side of the window or within the window. In fact, the status bar does not have to actually touch the window as long as it is clearly associated with the window. For example, the status bar could be provided within a small distance of each window, or it could be provided farther from a window and be coupled to the window by a line or the like.

It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A computer system for displaying a status bar for a window of an application program comprising:

a central processing unit (CPU);

read/write memory coupled to said CPU;

a computer screen coupled to said CPU;

means for selecting a status bar template from a plurality of predefined status bar templates, said selected status bar template to be associated with an application program, each of said plurality of status bar templates being able to provide a status bar for different application programs usable on said computer system and each template including a different number or type of active area;

means for providing a status bar from said selected status bar template independently of said application program and independently of an application window of said application program;

means for displaying said status bar on said computer screen such that said status bar is displayed external to said window and is visibly associated with a window of said application program which is also displayed on said computer screen, wherein said status bar is associated only with said application program and is always displayed when said window of said associated application program is displayed; and

means for displaying an active area within said status bar, said active area always being displayed within said status bar and being unable to be removed from said status bar while said status bar is displayed, said active area including an icon or a label.

2. A computer system for displaying a status bar as recited in claim **1** wherein said status bar is displayed in contact with said window.

3. A computer system for displaying a status bar as recited in claim **2** wherein a plurality of active areas are displayed within said status bar.

4. A computer system for displaying a status bar as recited in claim **2** wherein said active area includes said icon which, when activated, controls an operation of said application program.

5. A computer system for displaying a status bar as recited in claim **4** wherein said icon comprises a button which can be activated to control said operation.

6. A computer system for displaying a status bar as recited in claim **4** wherein said icon comprises a close box which can be activated to close said application program window.

7. A computer system for displaying a status bar as recited in claim **2** wherein said active area displays information provided by said application program.

8. A computer system for displaying a status bar as recited in claim **2** wherein said active area displays information not provided by said application program.

9. A computer system for displaying a status bar as recited in claim **8** wherein said information comprises the current time of day displayed in the form of a clock.

10. A computer system for displaying a status bar as recited in claim **2** wherein said active area controls a function which indirectly controls an operation of said application program.

11. A computer system for displaying a status bar as recited in claim **10** wherein said active area opens a new window on said computer screen.

12. A computer system for displaying a status bar as recited in claim **11** wherein said application window displays data provided by said application program, and wherein said new window comprises a menu of possible operations to be performed on said data displayed by said application program.

13. A computer system as recited in claim **1** further comprising means for moving said status bar template across said computer screen, wherein when said status bar template is moved, said application window associated with said status bar template is also moved across said computer screen.

14. A computer system as recited in claim **13** wherein said status bar is a selectable object, and wherein said means for moving said status bar includes a pointer for selecting objects displayed on said computer screen, wherein said status bar is moved by selecting said status bar and dragging said status bar across said computer screen with said pointer.

15. A computer system as recited in claim **1** wherein each of said status bar templates includes a different number or type of predefined active areas and customized active areas, where said predefined active areas are always included, unchanged, in a status bar provided frown an associated status bar template, and wherein said customized active areas are filled in said template by said application program.

16. A computer system as recited in claim **1** wherein each of said status bar templates is associated with types of application programs used on said computer system such that similar application programs are associated with status bar templates having similar action areas.

17. A method for providing a status bar for an application window on a computer screen of a computer system, the method comprising the steps of:

creating a plurality of status bar templates from each of which one or more status bars can be provided for different application programs usable on a computer system, each of said status bar templates having at least one area to be filled and having a different number or

type of said area from other status bar templates so that each template provides a status bar having a different appearance than status bars provided from other templates;

selecting one of said plurality of status bar templates to be associated with an application program having an associated application window displayed on said computer screen, such that only said selected status bar template is associated with said application window;

filling said area of said selected status bar template with an active area to create a status bar, said active area being filled in by said application program; and

displaying said status bar on a computer screen such that it is visually associated only with said application window and is external to said application window, said status bar always being displayed when said application window is displayed, wherein information associated with said application window is always displayed in said active area of said status bar when said status bar is displayed and wherein said information is always displayed in a same location within said status bar and is unable to be moved within said status bar.

18. A method for providing a status bar as recited in claim **17** wherein said status bar is displayed on said computer screen such that it contacts said application window.

19. A method for providing a status bar as recited in claim **18** wherein said status bar includes an active area which controls a function of said application program.

20. A method for providing a status bar as recited in claim **18** wherein said status bar includes an active area which displays information generated by said application program.

21. A method for providing a status bar as recited in claim **18** wherein said status bar includes an active area which displays information derived outside of said application program.

22. A method for providing a status bar as recited in claim **18** wherein said status bar includes an active area which controls a function outside of said application program.

23. A method for providing a status bar as recited in claim **17** wherein a plurality of active areas are filled in and displayed in said status bar, wherein at least one of said plurality of active areas is a predefined active area not filled in by said application program.

24. A method for providing a status bar as recited in claim **17** wherein a plurality of application windows, each associated with an application program, are displayed, and wherein a plurality of status bars corresponding to said plurality of application windows is displayed, wherein each status bar is visibly associated with a different one of said application windows.

25. A method as recited in claim **17**, wherein each of said status bar templates is previously associated with types of application programs used on said computer system such that templates having similar action areas are selected to be associated with similar application programs.

26. A method as recited in claim **17** wherein said active area filled into said area by said application program is a customized active area, and wherein at least one of said status bar templates includes at least one predefined active area that is always included in a status bar generated from said template having said predefined active area.

27. A method as recited in claim **26** wherein said customized active area is created by selecting an active area template from a plurality of predefined active area templates and generating said customized active area from said selected active area template.

28. A method as recited in claim **27** wherein said active area templates include a button template for providing a button active area on said status bar and a text template for providing a label active area on said status bar.

29. A computer system for displaying a status bar for a window of an application program comprising:

a central processing unit (CPU);

read/write memory coupled to said CPU;

a computer screen coupled to said CPU;

a tablet receptive to a stylus, said tablet being coupled to said CPU;

means for selecting a status bar template for an application program from a collection of available, predefined status bar templates, where said selecting occurs independently of said application program and independently of an application window of said application program, each of said status bar templates including a plurality of areas that are different in number or function from areas of other status bar templates in said collection, wherein each of said status bar templates can be used to provide a status bar for different application programs executed by said CPU and wherein each of said status bar templates is associated with a type of application program used on said computer system such that similar application programs are associated with particular status bar templates to provide a common status bar appearance for said similar application programs:

means for providing a status bar from said selected status bar template and displaying said status bar on said computer screen such that said status bar visibly contacts an application window of said application program which is also displayed on said computer screen, said status bar being displayed outside a perimeter of said application window, wherein said status bar is associated only with said application window and is only displayed when said application window of said associated application program is displayed; and

means for displaying a plurality of active areas in said areas of said selected template such that said active areas are displayed within said status bar, said plurality of active areas always being displayed and being unable to be removed from said computer screen when said status bar is displayed, said plurality of active areas including a button which can be selected by said stylus on said tablet to control an operation of said application program.

30. A computer system for displaying a status bar as recited in claim **29** wherein at least one of said plurality of active areas displays information provided by said application program.

31. A computer system for displaying a status bar as recited in claim **29** wherein at least one of said plurality of active areas displays information not provided by said application program.

32. A computer system for displaying a status bar as recited in claim **29** wherein said button, when selected, causes a new application program to start and a new application window associated with said new application program to be displayed on said computer screen, such that:

said means for selecting a status bar template selects a new status bar template from said collection of status templates for said new application window;

said means for displaying said status bar displays said new status bar such that it visibly contacts said new application window external to said new application window, wherein said new status bar is associated only with said new application window and is always dis-

played when said new application window is displayed; and

said means for displaying a plurality of active areas displays a new plurality of active areas within said new status bar, said plurality of new active areas always being displayed and being unable to be removed from said computer screen when said new status bar is displayed.

33. A computer system for displaying a status bar as recited in claim **29** wherein said status bar is a selectable object, and further comprising means for dragging said status bar across said computer screen when said stylus points to said status bar and is moved, wherein said associated application window is moved when said status bar is dragged.

34. A computer system as recited in claim **29** wherein at least one of said status bar templates includes a predefined action area in one of said areas, said predefined action area always being included in a status bar provided from said status bar template.

35. A computer system as recited in claim **34** wherein said status bar includes a customized active area which is filled into one of said areas by said application program.

36. A computer system as recited in claim **35** wherein said customized active area is provided by selecting an active area template from a plurality of available, predefined active area templates and generating said customized active area from said selected active area template.

37. A method for providing a status bar for an application window on a computer screen of a computer system, the method comprising the steps of:

creating a plurality of status bar templates from which one or more status bars can be provided, wherein a status bar can be provided for application programs using any of said status bar templates each of said status bar templates having a plurality of areas, and wherein each of said status bar templates has a different number or type of predefined active areas filled into at least one of said areas and at least one empty area, wherein said predefined active areas are always included in status bars derived from said status bar templates;

selecting one of said plurality of status bar templates to be associated with an application program having an associated application window displayed on said computer screen;

filling said empty area of said selected status bar template with a customized active area to create a status bar, said empty area being associated with and filled in by said application program, said status bar including said status bar template, said predefined active areas, and said customized active area; and

displaying said status bar on a computer screen such that it is visually associated only with said application window, said status bar always being displayed when said application window is displayed, wherein information associated with said application window is always displayed in said customized active area of said status bar when said status bar is displayed and wherein said active areas are always displayed in a same location within said status bar and are unable to be moved within said status bar.

38. A method as recited in claim **37** wherein types of application programs running on said computer system and particular status bar templates are associated with each other such that similar application programs are associated with templates having similar predefined action areas and empty areas to provide a common status bar appearance for said similar application programs.

39. A method as recited in claim **37** further comprising creating a plurality of active area templates, and wherein said customized active area is provided by selecting one of said active area templates from said plurality of active area templates and generating said customized active area from said selected active area template.

40. A method as recited in claim **39** wherein said active area templates include a button template for providing a button active area on said status bar and a text template for providing a label active area on said status bar.

* * * * *

US005617526A

United States Patent [19]

Oran et al.

[11] Patent Number: 5,617,526

[45] Date of Patent: Apr. 1, 1997

[54] **OPERATING SYSTEM PROVIDED NOTIFICATION AREA FOR DISPLAYING VISUAL NOTIFICATIONS FROM APPLICATION PROGRAMS**

[75] Inventors: **Daniel P. Oran**, Cambridge, Mass.; **Teresa A. Serrador**, Kirkland, Wash.; **Joseph D. Belfiore**; **George H. Pitt, III**, both of Redmond, Wash.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **355,398**

[22] Filed: **Dec. 13, 1994**

[51] **Int. Cl.**[6] **G06F 3/00**

[52] **U.S. Cl.** **395/326**; 395/348

[58] **Field of Search** 395/155, 159, 395/161

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,713,656	12/1987	Cliff et al.	340/723
5,237,653	8/1993	Noguchi et al.	395/158
5,261,044	11/1993	Dev et al.	395/159
5,333,256	7/1994	Green et al.	395/159
5,371,848	12/1994	Casey et al.	395/161
5,375,199	12/1994	Harrow et al.	395/159
5,471,399	11/1995	Tanaka et al.	395/161 X
5,483,631	1/1996	Nagai et al.	395/155

OTHER PUBLICATIONS

Cowart, R., "Mastering Windows 3.1," Sybex Publ., 1992, p. 909.

McClelland, Deke, *Macintosh® System* 7.1:*Everything You Need To Know*, Sybex Inc., San Francisco, Calif., 1992, pp. 37–38, 228.

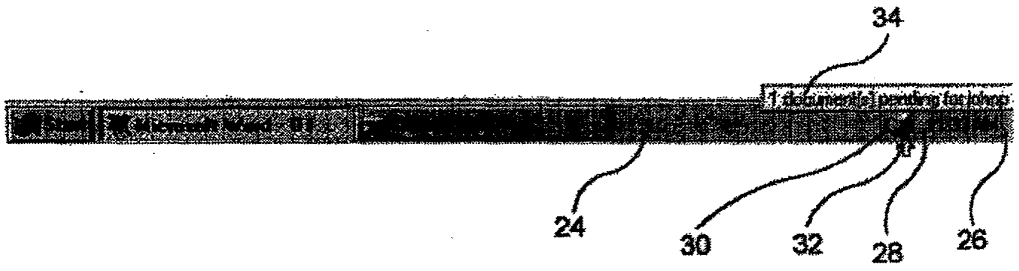
Knowledge Base Article describing Microsoft Mail 3.0 from Microsoft Development Network, 1994.

Primary Examiner—Raymond J. Bayerl
Assistant Examiner—A. Katbab
Attorney, Agent, or Firm—Seed and Berry LLP

[57] **ABSTRACT**

A system visual notification area is provided to display visual notifications to a user. The visual notifications provide visual cues to a user of the nature of the intended notification. The visual notifications may notify a user of events, status information, or other information. The visual notifications may be interactive so as to display information about a notification when a cursor is positioned to point at the notification. Moreover, the visual notifications may be interactive so as to provide a means for activating a program or displaying a menu to respond to the notification. The visual notifications are displayed without substantially interrupting execution of any currently active programs. Furthermore, the visual notifications may be displayed without the program that is provided to respond to the notifications having a currently active window.

41 Claims, 8 Drawing Sheets



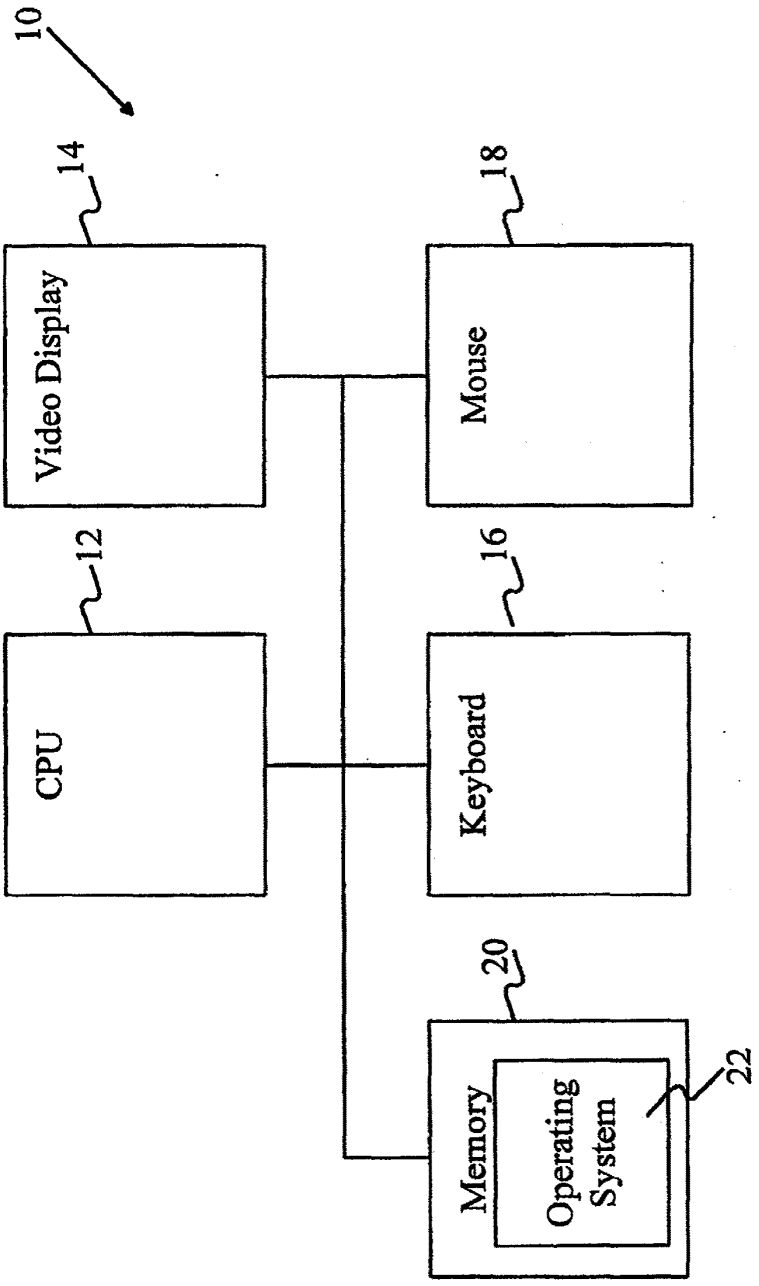


FIG. 1

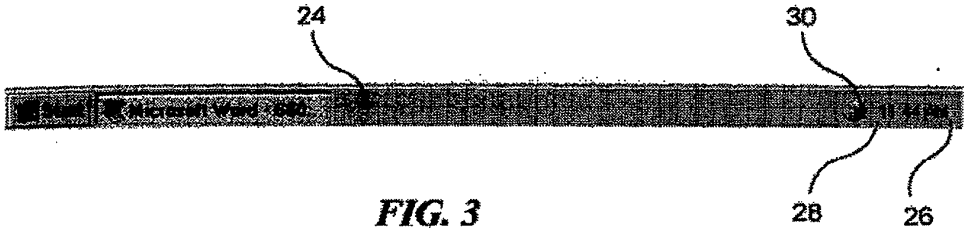


FIG. 3

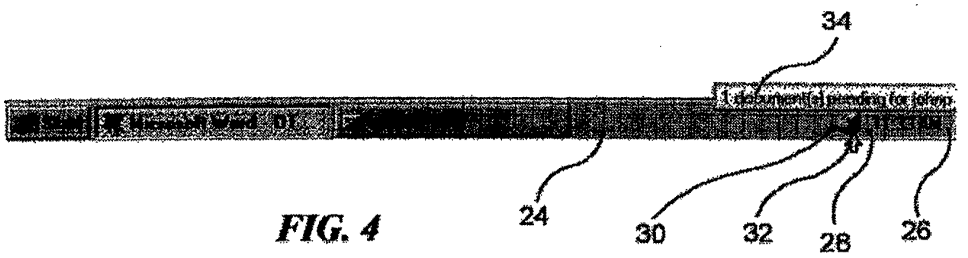


FIG. 4

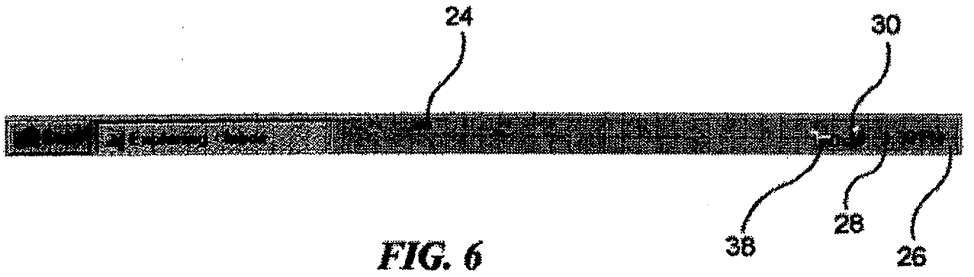


FIG. 6

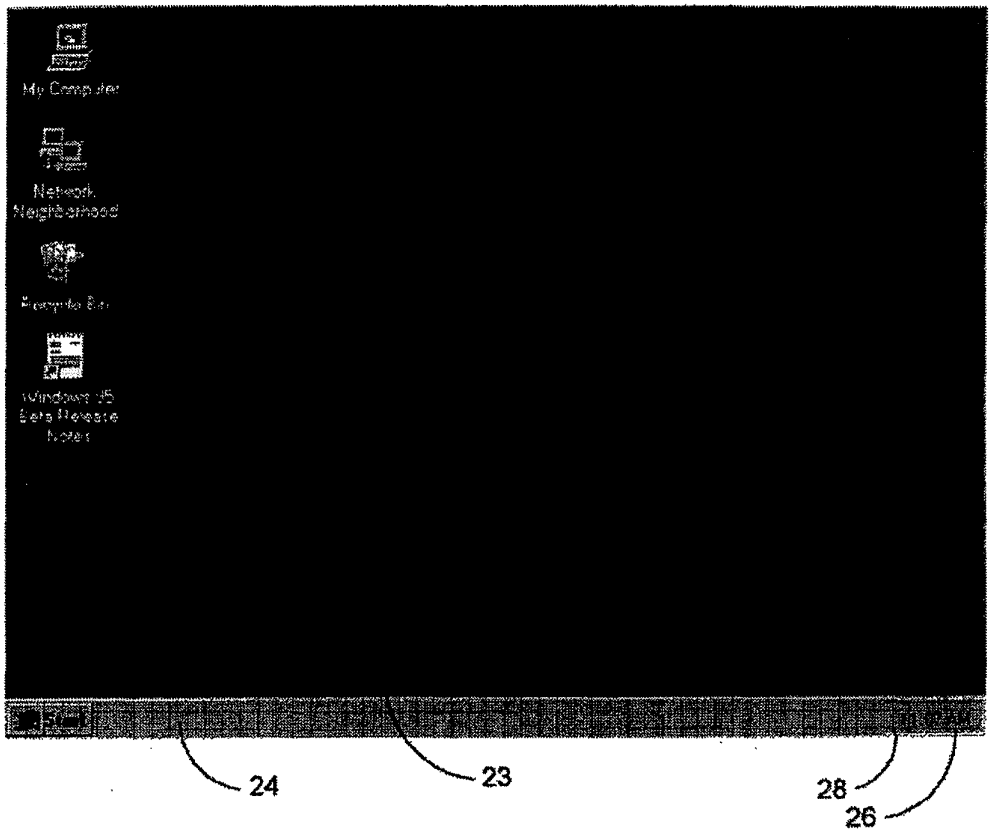


FIG. 2

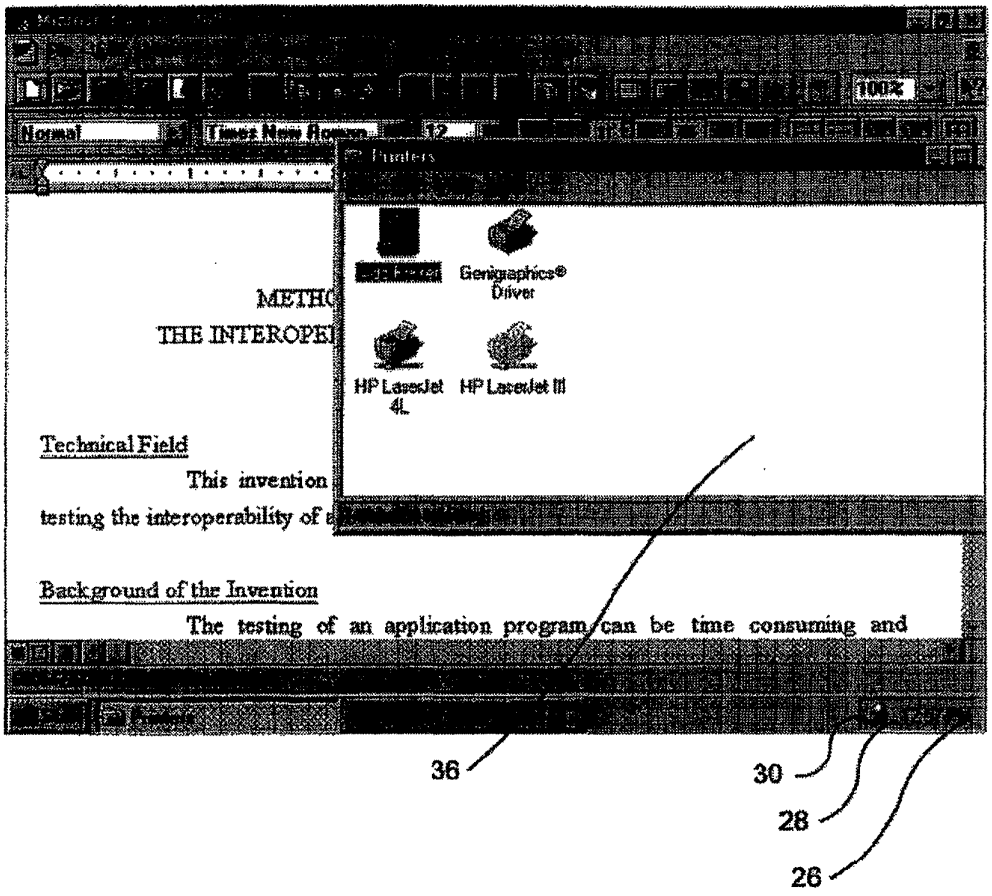


FIG. 5

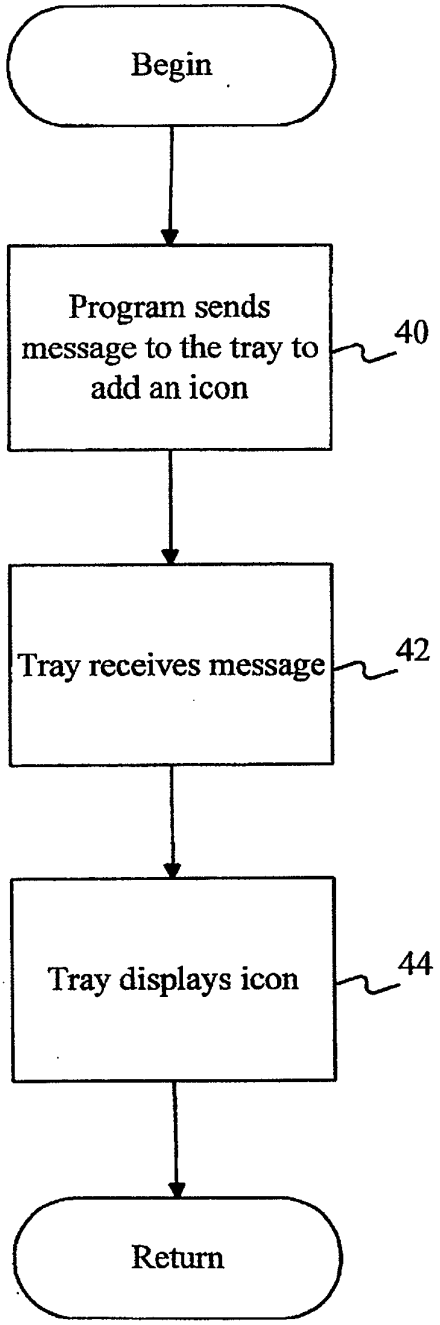


FIG. 7

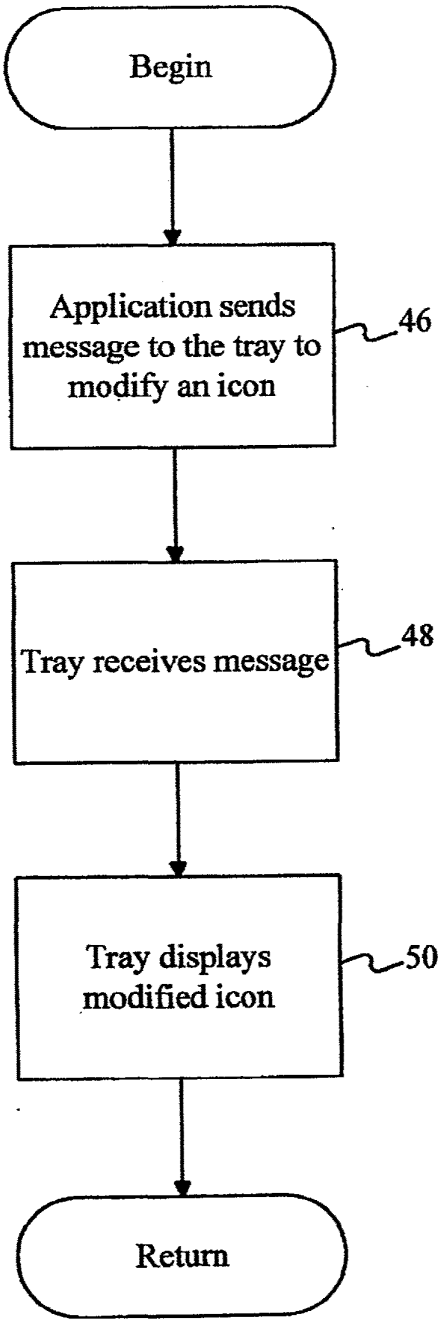

Begin
Application sends message to the tray to modify an icon
46
Tray receives message
48
Tray displays modified icon
50
Return

FIG. 8

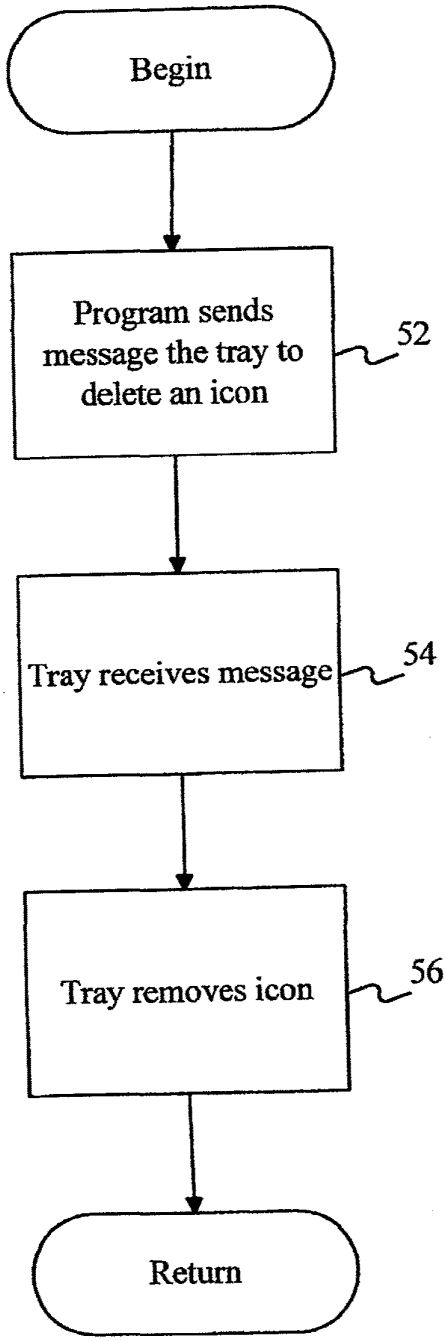


FIG. 9

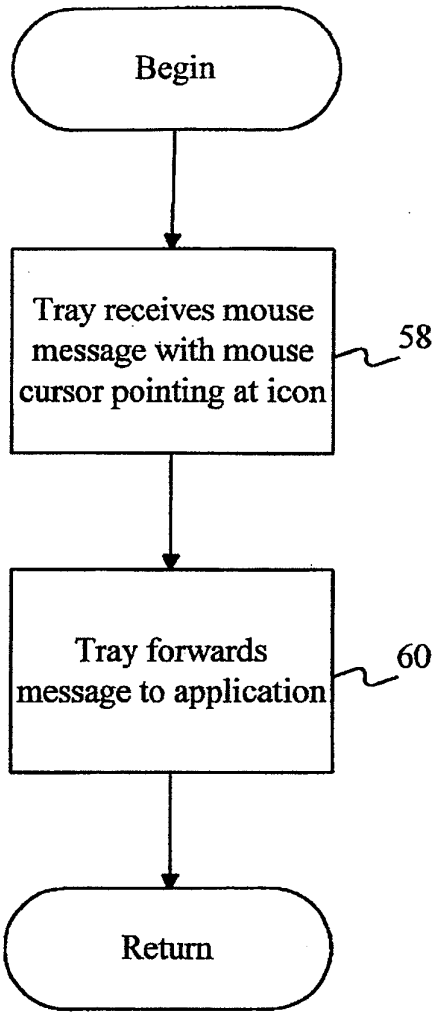

Begin
Tray receives mouse message with mouse cursor pointing at icon
58
Tray forwards message to application
60
Return

FIG. 10

OPERATING SYSTEM PROVIDED NOTIFICATION AREA FOR DISPLAYING VISUAL NOTIFICATIONS FROM APPLICATION PROGRAMS

TECHNICAL FIELD

The present invention relates generally to data processing systems and, more particularly, to notification of status information and events in a data processing system.

BACKGROUND OF THE INVENTION

Users of conventional operating systems are often uninformed of events outside of the currently active program. For example, in many conventional operating systems, it is difficult for a user to be informed that an electronic mail message has arrived. Systems that do provide some form of notification, such as a notification of the arrival of an electronic mail message, generally require that the associated program (i.e., the mail program) be visible and running in order for the notifications to arrive. It has also been difficult for users to obtain status information regarding resources in the computer system. For example, it is often difficult for users of portable computers to know whether the batteries in their portable computers need to be recharged and will soon be out of power.

SUMMARY OF THE INVENTION

The above-described difficulties encountered with conventional systems are overcome by the present invention by providing a system-wide visual notification area. In accordance with a first aspect of the present invention, a method is practiced on a computer system that has an output device and that runs an operating system. The operating system provides a graphical user interface on the output device. A portion of the graphical user interface is designated as a notification area for displaying notifications. Each notification includes a graphical object. A system tool is provided in the computer system to display a notification in the notification area. An application program is run on the computer system, and the notification is displayed in the notification area using the system tool, without substantially interrupting the running of the application program.

In accordance with another aspect of the present invention, a first program is run on a computer system that provides a user interface on the output device. An application program is also run on the computer system. A system tool is provided that designates a portion of the user interface as a notification area that is reserved for displaying visual notifications to a user. A third program is also run on the computer system. This third program provides a request to display a selected visual notification in the notification area. In response to the request, the selected visual notification is displayed in the notification area using the system tool without substantially interrupting the running of the second program.

In accordance with yet another aspect of the present invention, a computer system includes an output device and a processor for running programs. The processor runs an application program and an operating system. The operating system displays a graphical user interface on the output device. The processor also runs a system tool that displays visual notifications for a user in a designated notification area on the graphical user interface. The system tool displays the visual notifications without substantially interrupting the running of the application program on the user interface.

BRIEF DESCRIPTION OF THE DRAWINGS

A preferred embodiment of the present invention will be described below with reference to the following figures.

FIG. **1** is a block diagram of a computer system that is suitable for practicing the preferred embodiment to the present invention.

FIG. **2** is an example showing a virtual desktop and taskbar notification area in accordance with the preferred embodiment to the present invention.

FIG. **3** shows an example of an icon being displayed in the taskbar notification area in the preferred embodiment of the present invention.

FIG. **4** shows an example of a tool tip being generated for an icon in the taskbar notification area in accordance with the preferred embodiment of the present invention.

FIG. **5** shows an example of a window being opened in response to double-clicking a mouse on an icon in the taskbar notification area per the preferred embodiment of the present invention.

FIG. **6** shows an example of a taskbar notification area displaying multiple icons simultaneously in accordance with the preferred embodiment of the present invention.

FIG. **7** is a flow chart illustrating the steps that are performed to display an icon in the taskbar notification area in accordance with the preferred embodiment to the present invention.

FIG. **8** is a flow chart illustrating the steps that are performed to modify an icon that is currently displayed in the taskbar notification area per the preferred embodiment to the present invention.

FIG. **9** is a flow chart illustrating the steps that are performed to delete an icon that is currently displayed in a taskbar notification area in accordance with the preferred embodiment of the present invention.

FIG. **10** is a flow chart illustrating the steps that are performed to notify application programs of mouse activities relative to an icon in the taskbar notification area per the preferred embodiment to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The preferred embodiment of the present invention provides a taskbar notification area at a predefined location on a user interface wherein icons may be displayed to inform a user of status information and events. This dedicated taskbar notification area is always visible and cannot be readily obscured by visible non-minimized windows and the like. The icons are placed within the taskbar notification area without disrupting work flow of a user and may be placed in the taskbar notification area without the need for an associated program that requests the notifications having a correctly active window. The icons displayed within the taskbar notification area may be interactive in that tool tip information may be provided in response to the user pointing at the icon with the mouse and the application may perform activities, such as the opening of an application window, in response to the user using the mouse to double-click on the icon.

FIG. **1** is a block diagram of a computer system **10** that is suitable for practicing the preferred embodiment of the present invention. The computer system **10** includes a central processing unit (CPU) **12** that is coupled to a number of peripheral devices. The peripheral devices include a video

display **14**, a keyboard **16**, and a mouse **18**. The CPU **12** is also connected to a memory **20** that holds a copy of an operating system **22**.

In the preferred embodiment of the present invention, the taskbar notification area is provided as part of the shell of the operating system **22**. As such, the code for implementing the taskbar notification area is included within the operating system **22**. Although the preferred embodiment of the present invention will be described with reference to an implementation wherein the taskbar notification area is implemented as part of the operating system **22**, those skilled in the art will recognize that the taskbar notification area may be implemented as part of an application program or as a separate system resource that interacts with the operating system **22**. Those skilled in the art will also appreciate that the present invention may be practiced in computer system configurations that differ from that shown in FIG. **1**. The computer system of FIG. **1** is intended to be merely illustrative.

For purposes of the discussion below, it will be assumed that the operating system **22** is the "WINDOWS" 95 operating system from Microsoft Corporation of Redmond, Wash. Those skilled in the art will appreciate that the present invention may also be practiced with other operating systems.

FIG. **2** depicts an example of the user interface that is displayed by shell of the operating system **22**. The user interface includes a virtual desktop **23** with a taskbar **24** located at the bottom of the virtual desktop. The taskbar **24** is like that described in co-pending patent application entitled "Taskbar and Start Menu," Ser. No. 08/478,490, which is assigned to a common assignee with the present application and which was filed on even date herewith. The taskbar **24** includes a dedicated area **26** for showing a clock. In the example in FIG. **2** the clock displays a time of 6:54 PM. Directly adjacent to the clock area **26** is the taskbar notification area **28**. In FIG. **5** the taskbar notification area is shown as being to the left of the clock. Nevertheless, it should be appreciated that the taskbar notification area may appear at other locations relative to the clock. In the example shown in FIG. **2**, no icon is currently displayed within the taskbar notification area **28**. As will be described in more detail below, the size of the taskbar notification area **28** increases to accommodate the space requirements needed to display icons and shrinks to conserve space when space requirements for displaying the icons decreases.

In order to fully appreciate the operation of the taskbar notification area **28** in the preferred embodiment of the present invention, it is helpful to consider an example wherein the taskbar notification area is used. Suppose that a user wishes to print a word processing document. When the user begins to print the word processing document, the taskbar notification area **28** appears as shown in FIG. **3**. In particular, a printer icon **30** is displayed within the taskbar notification area **28**. The printer icon **30** is displayed without substantially interrupting the execution of the currently active application. The taskbar notification area **28** has been expanded from its previous size (shown in FIG. **2**) to accommodate the display of the printer icon **30**. The printer icon **30** advises a user that at least one document is currently being printed. The printer icon **30** provides a visual cue that indicates the nature of the notification.

Those skilled in the art will appreciate that the present invention is not limited to displaying icons in the taskbar notification area, rather any type of graphical object or visual notification may be displayed in the taskbar notification area. Moreover, these notifications may be accompanied by sound notifications, such as a chime.

If the user uses mouse **18** to position a mouse cursor **32** to point at the printer icon **30** within the taskbar notification area **28**, a tool tip **34** will be displayed as shown in FIG. **4**. The tool tip **34** is displayed solely in response to the mouse cursor **32** being positioned to point at the printer icon **30**. In the example shown in FIG. **4**, the tool tip indicates the number of documents that are currently pending on the printer for the user. The tool tip that is displayed is like that described in co-pending patent application entitled "Timing and Velocity Control for Displaying Graphical Information," Ser. No. 08/260,558, filed on Jun. 16, 1994, and assigned to a common assignee with the present application. The nature of the information provided by the tool tip depends upon the icon shown within the taskbar notification area. In general, the tool tip provides information about the notification. The tool tip **34** provides a first level of interactivity for the icons in the taskbar notification area **28**.

A second level of interactivity for the icons in the taskbar notification area is provided when the mouse cursor **32** is positioned to point at an icon **30** within the taskbar notification area **28**, and then the user double-clicks a designated one of the mouse buttons. The application program associated with the icon receives a message that the user has double-clicked on the icon and then determines what actions to take in response to the double-clicking. In the case of the printer icon **30**, the print manager is the application that receives the double-click message and, in response, it opens the print manager window **36**. More generally, in the preferred embodiment the associated application provides code for notifying an application associated with an icon in the taskbar notification area **28** of activities that are performed relative to the icon. These activities may include events like single clicking on the icon, double-clicking on the icon or the mouse cursor being positioned to point at the icon. The application associated with the icon receives this message and determines what activities to perform.

The taskbar notification area **28** is not a static area. It grows and shrinks to accommodate the number of icons that are currently displayed within it. The taskbar notification area **28** is a window that may be resized and relocated. For example, as shown in FIG. **6**, the taskbar notification area **28** has been expanded from its size in FIG. **3** to accommodate the display of both printer icon **30** and mail icon **38**. The mail icon **38** indicates that an electronic mail message has arrived for a user. Further suppose the user then looks at the electronic mail message. In such a case, the taskbar notification area **28** shrinks in size to display only the printer icon **30** as shown in FIG. **5**.

The above examples have related to instances wherein the notifications notify a user of events. It should also be appreciated that the notifications may convey status information, such as the status of a modem or the status of a user's batteries in a portable computer.

The above discussion has focused on the description of the present invention from the user's perspective. The discussion below will now focus on the implementation of the preferred embodiment of the present invention at a code level.

FIG. **7** shows a flowchart of the steps that are performed in order to display an icon within the taskbar notification area **28**. Initially, an application program sends a message to a process run within the operating system, known as the taskbar, which is responsible for displaying and managing the taskbar **24**. The taskbar's responsibilities also include displaying and managing the taskbar notification area **28**.

The application program sends the message by calling the Shell_NotifyIcon() function as defined by the "WINDOWS" 95 operating system. This function sends a message to the taskbar that includes several parameters. A first of the parameters of the message identifies whether a user wishes to add an icon, modify an icon, or delete an icon from the taskbar notification area **28**. In the present example, the message indicates that the user wishes to add an icon to the taskbar notification area. A second parameter passed in the message specifies the size of the information that follows the parameter. A third parameter is a value for a window handle to the application program window that is sending the message. The window handle is a numerical identifier that uniquely identifies the window.

A fourth parameter passed in this message is the unique identifier that is application specific. A single application may have multiple unique identifiers associated with it. The interpretation of these identifiers is at the discretion of the application. Additional parameters passed in the message include flags that identify whether fields in the message that follow the flags parameter are valid or not. A callback message identifier parameter is passed as part of the message. The callback message identifier is an integer that uniquely identifies a message as a callback message. This callback message identifier is used by the taskbar to send callback messages to the target application window. The two final parameters passed in the message are a handle to the icon to show in the taskbar notification area and a string that serves as a tool tip message.

The taskbar receives the message that encapsulates the parameters discussed above (step **42** in FIG. **7**). The taskbar then uses the handle for the icon that was passed as part of the message from the application program to retrieve and display the icon within the taskbar notification area **28** without substantially interrupting the execution of the active application (step **44**).

Consider an example wherein a mail program wishes to advise the taskbar that it has received a new electronic mail message. In such a case, the mail program sends a message to the taskbar asking the taskbar to add an icon that displays a single envelope within a mail slot. The handle to the icon showing the single envelope in the mail slot is passed as part of the message that is sent to the taskbar. The taskbar then uses the handle provided by the mail program to display the icon in the taskbar notification area **28**.

The steps that are performed when an application program wishes to modify the icon shown within the taskbar notification area **28** are shown in FIG. **8**. The application initially sends a message to the taskbar that specifies it wishes to modify an icon (step **46**). For example, suppose that the mail program wishes to change the icon to one that displays a mail slot with multiple envelopes to indicate that the user has now received multiple electronic mail messages. In such a case, the mail program sends a message to the taskbar indicating that it wishes to modify the icon and passes the handle to the new icon to be displayed. The taskbar receives the message (step **48**), and the taskbar retrieves the modified icon and displays it in the taskbar notification area **28** (step **50**).

The deletion of icons in the taskbar notification area **28** is handled in a similar fashion. As shown in FIG. **9**, an application program sends a message to the taskbar to delete an icon using the Shell_NotifyIcon() function as described above (step **52**). Suppose that a print job is done and the print manager wishes to remove the printer icon **30** (FIG. **5**) from the taskbar notification area **28**. In such a case, the print manager sends a message to the taskbar to delete the printer icon **30**. The taskbar receives the message (step **54**), and then, the taskbar removes the icon so that it is no longer displayed in the taskbar notification area (step **56**). In the example case of the printer icon **30**, the taskbar receives a message from the print manager and removes the printer icon from the taskbar notification area **28**.

The notification area is part of the taskbar **24**, which constitutes a separate window. The taskbar receives mouse messages when the mouse cursor is positioned over the taskbar window, including the taskbar notification area. The taskbar is responsible for notifying the application associated with icons in the taskbar notification area **28** of mouse activities. Thus, when the mouse cursor is positioned to point at one of the icons displayed within the taskbar notification area **28**, the taskbar performs the steps shown in FIG. **10**. In particular, the taskbar receives a mouse message that specifies the coordinates of the mouse cursor and processes these coordinates to determine that the mouse cursor is currently pointing at one of the icons shown within the taskbar notification area **28** (step **58**) in FIG. **10**. The taskbar then sends a callback message to the application program associated with the icon to inform the application of the mouse activity. In particular, the taskbar may forward messages that were originally sent to the taskbar notification area. In this regard, the taskbar acts as a parent window. Thus, an application program may be informed of current mouse coordinates, whether the left, middle, or right button on the mouse is depressed or up, whether the mouse has moved, whether one of the mouse buttons has been clicked, and whether a double-click of one of the mouse buttons has occurred. Using this mechanism, the taskbar advises an application program of when an icon has been double-clicked on and the icon then decides what activities to perform (e.g., opening the application window).

Those skilled in the art will appreciate that the icons displayed within the figures in the above description are not intended to be exhaustive of the icons that may be displayed within the taskbar notification area. Moreover, those skilled in the art will appreciate that the examples given above are intended to be merely illustrative and that other behaviors may be practiced by application programs.

While the present invention has been described with reference to a preferred embodiment thereof, those skilled in the art will appreciate that various changes in form and detail may be made without departing from the intended scope of the invention as defined in the appended claims.

We claim:

1. In a computer system having an output device and running an operating system having a shell that provides a shell graphical user interface on the output device, a method comprising the steps of:

designating a portion of the shell graphical user interface as a notification area for displaying notifications, each said notification including a graphical object;

running a first application program on the computer system; and

displaying a notification from a second application in the notification area with the shell, without substantially interrupting the running of the first application program.

2. The method of claim **1** wherein the graphical object included in the notification is an icon and wherein the step of displaying the notification comprises the step of displaying the icon without substantially interrupting the running of the first application program.

3. The method of claim 1 wherein the step of displaying the notification comprises the step of displaying a notification in the notification area using the system without substantially interrupting the running of the first application program to notify a user of an event.

4. The method of claim 1 wherein the step of displaying the notification comprises the step of displaying a notification in the notification area using the system without substantially interrupting the running of the first application program to convey status information to a user.

5. The method of claim 1, further comprising the step of displaying a second notification in the notification area from a third application program using the system tool without interrupting the running of the first application program such that the second notification is displayed in the notification area along with the first notification that is already displayed.

6. The method of claim 1 wherein the computer system further comprises an input device for manipulating a cursor on the output device in response to user actions and wherein the method further comprises the step of displaying information about the notification solely in response to the cursor pointing at the notification.

7. The method of claim 1 wherein the computer system further comprises an input device for use by a user and wherein the method further comprises the step of activating a program to respond to the notification in response to the user using the input device to perform activities on the notification.

8. The method of claim 7 wherein the input device is a mouse and the activities are double-clicking with the mouse on the notification.

9. The method of claim 1, further comprising the step of removing the notification from the notification area when the notification is no longer required.

10. The method of claim 1 wherein the system tool is part of the operating system.

11. In a computer system having an operating system and an output device, a method comprising the steps of:

running a first program on the computer system that provides a taskbar on the output device and a second program that is an application program, said taskbar by said operating system;

designating a portion of the taskbar as a notification area that is reserved for displaying visual notifications to a user;

running a third program on the computer system that provides a request to display a selected visual notification in the notification area; and

in response to the request, displaying the selected visual notification in the notification area by said operating system without substantially interrupting the running of the second program.

12. The method of claim 11 wherein the first program is an operating system and the step of running the first program comprises the step of running the operating system on the computer system.

13. The method of claim 12 wherein the system tool is provided by the operating system.

14. The method of claim 13 wherein the step of using the system resource to display the second visual notification further comprises the step of expanding the notification area to facilitate displaying the second visual notification along with the selected visual notification.

15. The method of claim 11, further comprising the steps of:

running a fourth program on the computer system that provides a request to display a second visual notification in the notification area;

in response to the request from the fourth program, using the system resource to display the second visual notification along with the selected visual notification in the notification area without substantially interrupting the running of the second program.

16. The method of claim 11 wherein the computer system further comprises an input device that a user may use to manipulate a cursor on the output device and wherein the method further comprises the step of displaying information about the selected visual notification solely in response to the cursor pointing at the selected visual notification in the notification area.

17. The method of claim 11 wherein the selected visual notification provides a user with a visual cue about a nature of the notification.

18. The method of claim 11 wherein the selected visual notification is an icon and the step of displaying the selected visual notification comprises the step of displaying the icon in the notification area.

19. The method of claim 11 wherein the step of running the third program on the computer system is performed without displaying an active window for the third program on the output device.

20. The method of claim 11 wherein the computer system further comprises an input device that the user may use to perform activities on the selected visual notification and in response to the user using the input device to perform activities on the selected icon, opening an active window on the output device for the third program.

21. The method of claim 11, further comprising the step of displaying additional visual notifications in the notification area using the system tool wherein each additional visual notification is unique and each additional visual notification provides a visual cue of a nature of the notification.

22. The method of claim 11 wherein the selected visual notification notifies the user of an event.

23. The method of claim 11 wherein the selected visual notification conveys status information to the user.

24. In a computer system having an output device and running an operating system and a first application program, a method comprising the computer-implemented steps of:

displaying a taskbar by the operating system on the output device;

designating a portion of the taskbar as a notification area for displaying notifications wherein each said notification includes a graphical object; and

displaying a first notification in the notification area at a request of the first application program.

25. The method of claim 24 wherein the computer system runs a second application program and wherein the method further comprises the step of displaying a second notification concurrently with the first notification in the notification area in response to a request by the second application program.

26. The method of claim 24, further comprising the step of displaying an additional notification in the notification area so that both the first notification and the additional notification are displayed in the notification area in response to a request by the first application program and the first notification differs from the additional notification.

27. The method of claim 24 wherein the computer system includes a second application program that does not currently have a non-closed window and wherein the method further comprises the step of displaying a second notification in the notification area in response to a request by the second application program.

28. The method of claim 24 wherein the first application program provides the graphical object of the first notification to the operating system for display in the notification area.

29. In a computer system having an output device and running an operating system having a shell that provides a shell graphical user interface on the output device, a computer-readable storage medium holding instructions for performing a method comprising the steps of:

designating a portion of the shell graphical user interface as a notification area for displaying notifications, each said notification including a graphical object;

running a first application program on the computer system; and

displaying a notification from a second application in the notification area with the shell without substantially interrupting the running of the first application program.

30. The computer-readable storage medium of claim 29 wherein the graphical object included in the notification is an icon and wherein the step of displaying the notification comprises the step of displaying the icon without substantially interrupting the running of the first application program.

31. The computer-readable storage medium of claim 29 wherein the step of displaying the notification comprises the step of displaying a notification in the notification area using the system without substantially interrupting the running of the first application program to notify a user of an event.

32. The computer-readable storage medium of claim 29 wherein the step of displaying the notification comprises the step of displaying a notification in the notification area using the system without substantially interrupting the running of the first application program to convey status information to a user.

33. In a computer system having an operating system and an output device, a computer-readable storage medium holding instructions for performing a method comprising the steps of:

running a first program on the computer system that provides a taskbar on the output device and a second program that is an application program, said taskbar by said operating system;

designates a portion of the taskbar as a notification area that is reserved for displaying visual notifications to a user;

running a third program on the computer system that provides a request to display a selected visual notification in the notification area; and

in response to the request, displaying the selected visual notification in the notification area by said operating tool without substantially interrupting the running of the second program.

34. The computer-readable storage medium of claim 33 wherein the first program is an operating system and the step of running the first program comprises the step of running the operating system on the computer system.

35. The computer-readable storage medium of claim 34 wherein the method further comprises the steps of:

running a fourth program on the computer system that provides a request to display a second visual notification in the notification area; and

in response to the request from the fourth program, using the system resource to display the second visual notification along with the selected visual notification in the notification area without substantially interrupting the running of the second program.

36. The computer-readable storage medium of claim 35 wherein the step of using the system resource to display the second visual notification further comprises the step of expanding the notification area to facilitate displaying the second visual notification along with the selected visual notification.

37. In a computer system having an output device and running an operating system and a first application program, a computer-readable storage medium holding instructions for performing a method comprising the computer-implemented steps of:

displaying a taskbar by the operating system on the output device;

designating a portion of the taskbar as a notification area for displaying notifications wherein each said notification includes a graphical object; and

displaying a first notification in the notification area at a request of the first application program.

38. The computer-readable storage medium of claim 37 wherein the computer system runs a second application program and wherein the method further comprises the step of displaying a second notification concurrently with the first notification in the notification area in response to a request by the second application program.

39. The computer-readable storage medium of claim 37, further comprising the step of displaying an additional notification in the notification area so that both the first notification and the additional notification are displayed in the notification area in response to a request by the first application program and the first notification differs from the additional notification.

40. The computer-readable storage medium of claim 37 wherein the computer system includes a second application program that does not currently have a non-closed window and wherein the method further comprises the step of displaying a second notification in the notification area in response to a request by the second application program.

41. The computer-readable storage medium of claim 37 wherein the first application program provides the graphical object of the first notification to the operating system for display in the notification area.

* * * * *

US005644334A

United States Patent [19]

Jones et al.

[11] Patent Number: 5,644,334

[45] Date of Patent: Jul. 1, 1997

[54] **STATUS INDICATORS OF AN IMPROVED GRAPHICAL USER INTERFACE**

[75] Inventors: **Jeremy A. Jones**, Arlington; **Neil L. Mayle**; **Paige K. Parsons**, both of Cambridge; **Andrew L. M. Shalit**, Somerville; **Steven H. Strassmann**, Cambridge, all of Mass.

[73] Assignee: **Apple Computer, Inc.**, Cupertino, Calif.

[21] Appl. No.: **613,530**

[22] Filed: **Mar. 11, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 245,877, May 19, 1994, abandoned.

[51] **Int. Cl.**[6] **G09B 1/06**

[52] **U.S. Cl.** **345/119**; 345/127; 345/157; 395/326; 395/340

[58] **Field of Search** 345/146, 118, 345/119, 120, 112, 127, 131; 395/155, 156, 157, 139

[56] **References Cited**

U.S. PATENT DOCUMENTS

Re. 32,632	3/1988	Atkinson	345/146
4,278,973	7/1981	Hughes et al.	345/120
4,622,545	11/1986	Atkinson	345/191
4,831,556	5/1989	Oono	345/119
4,881,179	11/1989	Vincent	345/118
4,888,712	12/1989	Barkans et al.	345/118
4,931,783	6/1990	Atkinson	345/146
5,065,347	11/1991	Pajak et al.	395/159
5,339,391	8/1994	Wroblewski et al.	345/157

OTHER PUBLICATIONS

IBM Technical Disclosure Bulleton, vol. 35, No. 7, Dec. 1992 New York, pp. 285–286, Selecting of Usable Font Size for the Host Sessions.

IBM Technical Disclosure Bulletin, vol. 34, No. 7A, Dec. 1991 New York, pp. 431–433, Providing a User Customized Details View.

Patent Abstracts of Japan, vol. 018 No. 587 (P1824), 10 Nov. 1994, & JP,A,06, 215095 (CSK Corp.) 5 Aug. 1994.

Guide to MacApp Tools, Developer Technical Publications, Apple Computer, Inc. 1992, pp. 1–162.

Languages for Developing User Interfaces, A Component Architecture for Personal Computer Software, Smith et al., pp. 31–56.

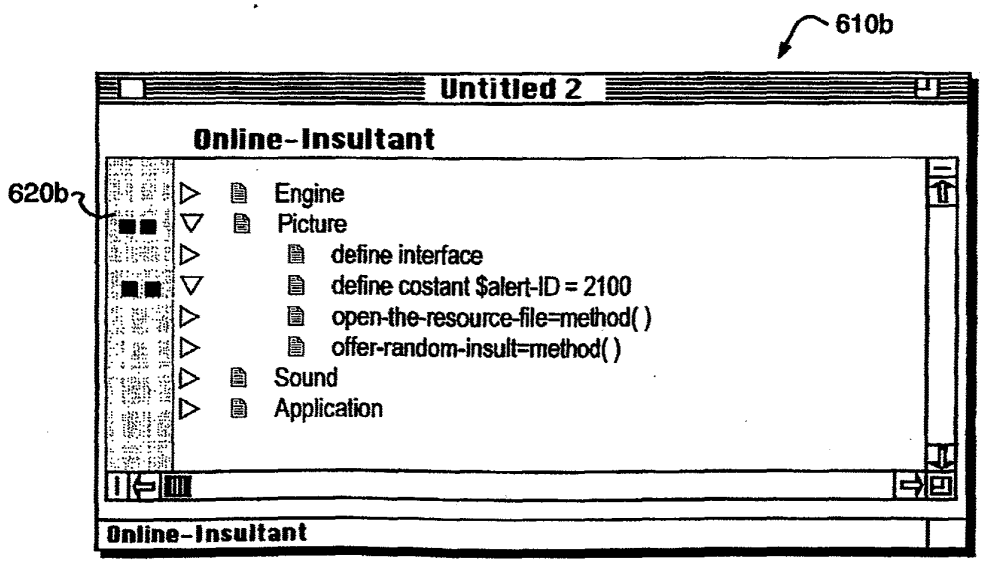
Desktop Sparc, Sun Systems User's Guide, Copyright 1990, Sun Microsystems, Inc., p. 143.

Primary Examiner—Kee M. Tung
Assistant Examiner—Doom Chow
Attorney, Agent, or Firm—Cesari and McKenna

[57] **ABSTRACT**

An improved graphical user interface comprises novel status indicators pertaining to state attributes associated with objects displayed on a display screen of a computer system. These status indicators are preferably portrayed on a window pane of a display screen as distinct visual cues and are located adjacent to their associated objects to provide a customizable browser framework to a user. A dynamically-adjustable side bar panel provides a designated area within each pane for displaying the status indicators.

16 Claims, 10 Drawing Sheets



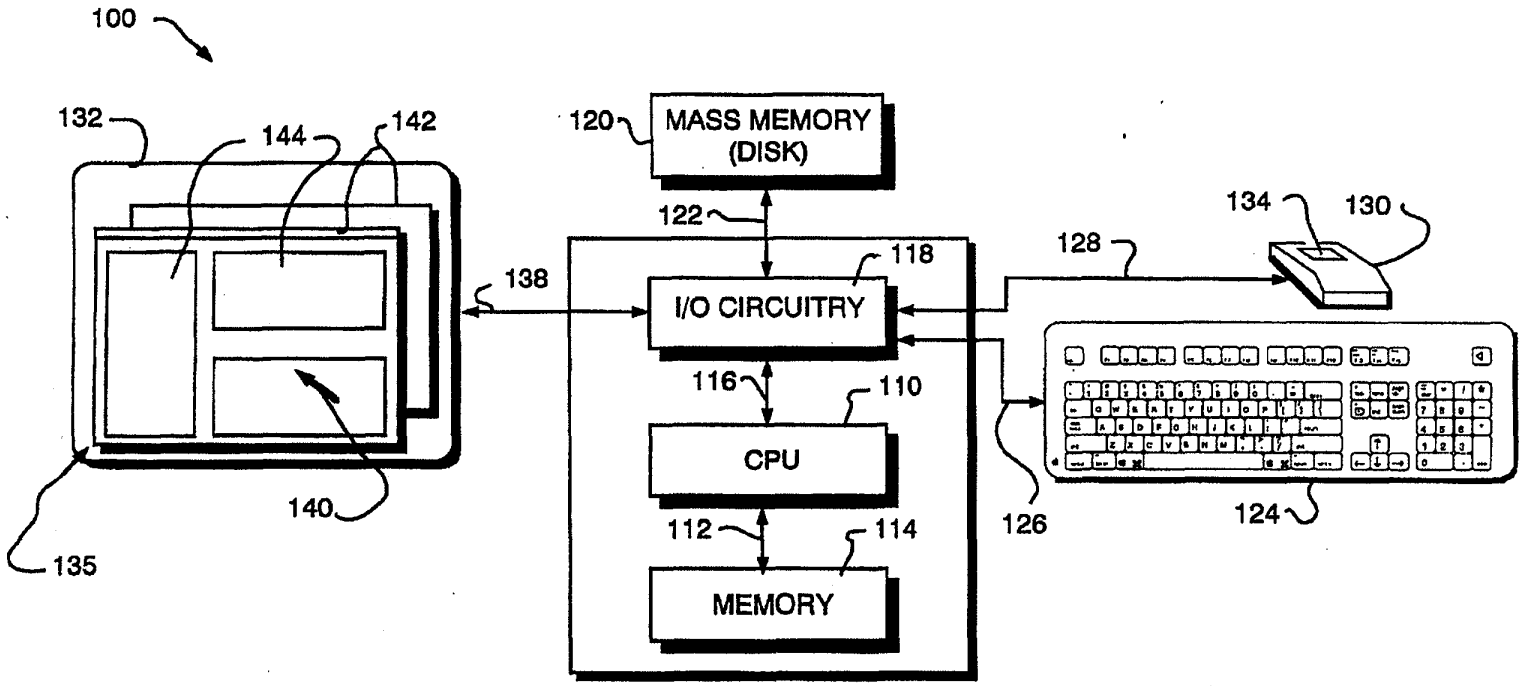

100
132
144
142
120
MASS MEMORY (DISK)
122
134
130
128
118
138
I/O CIRCUITRY
116
110
CPU
112
114
MEMORY
140
135
126
124

FIG. 1

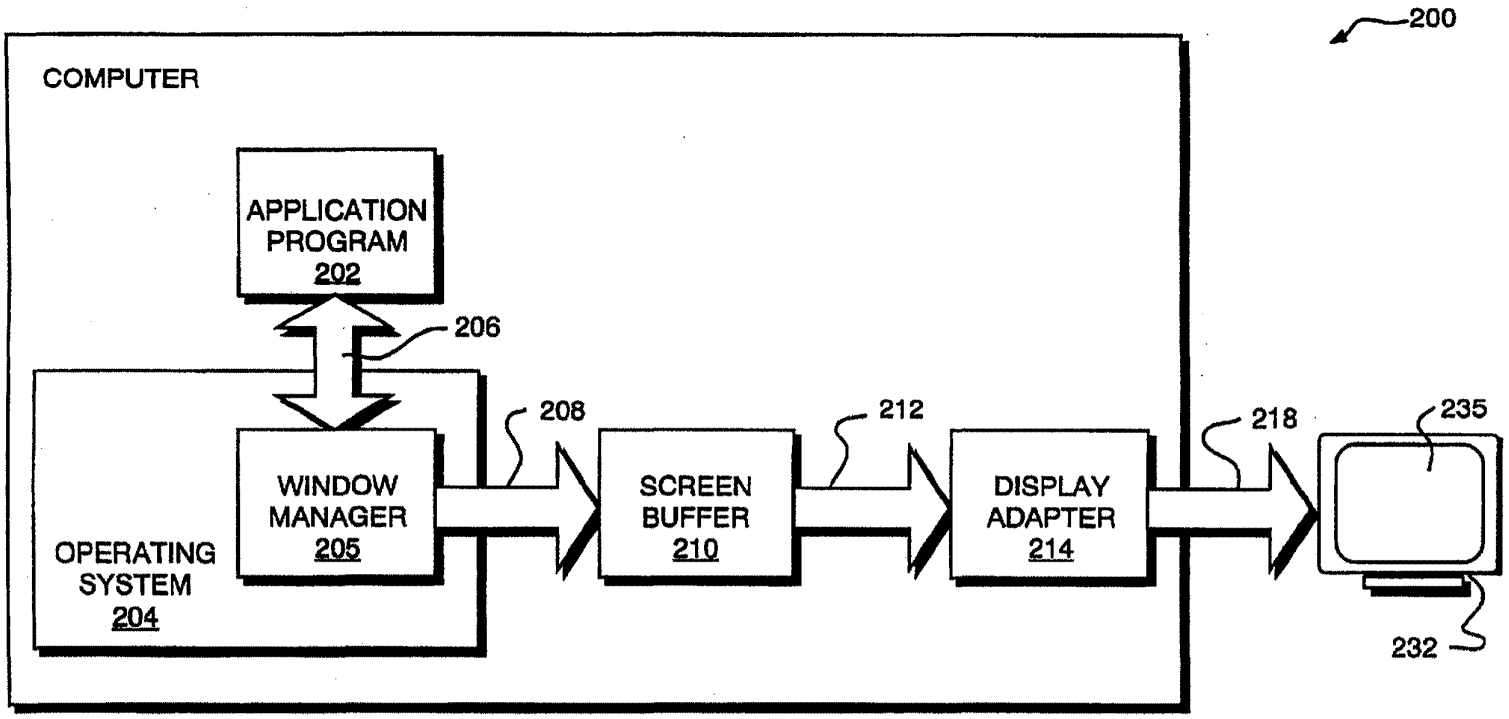


FIG. 2

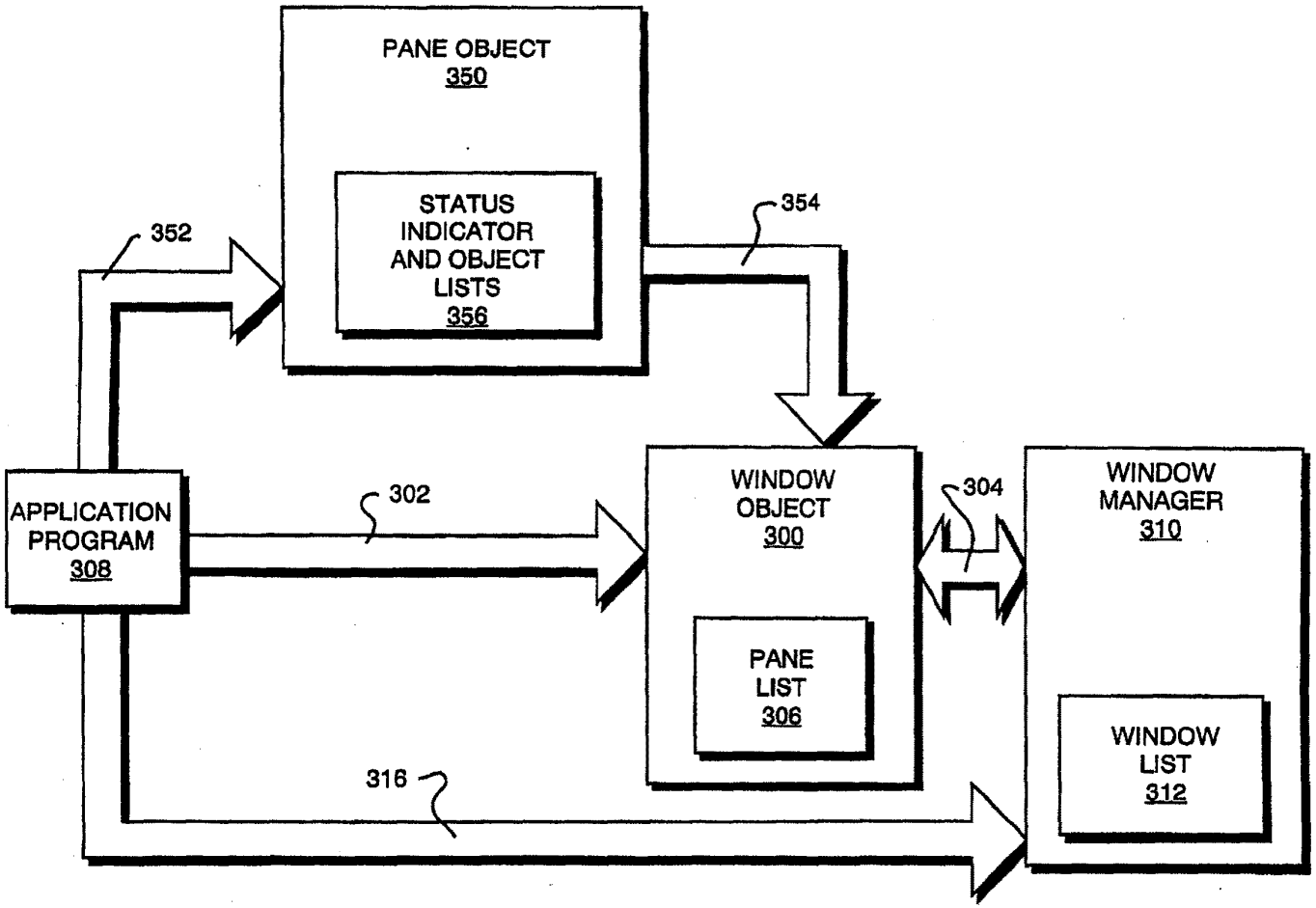


FIG. 3

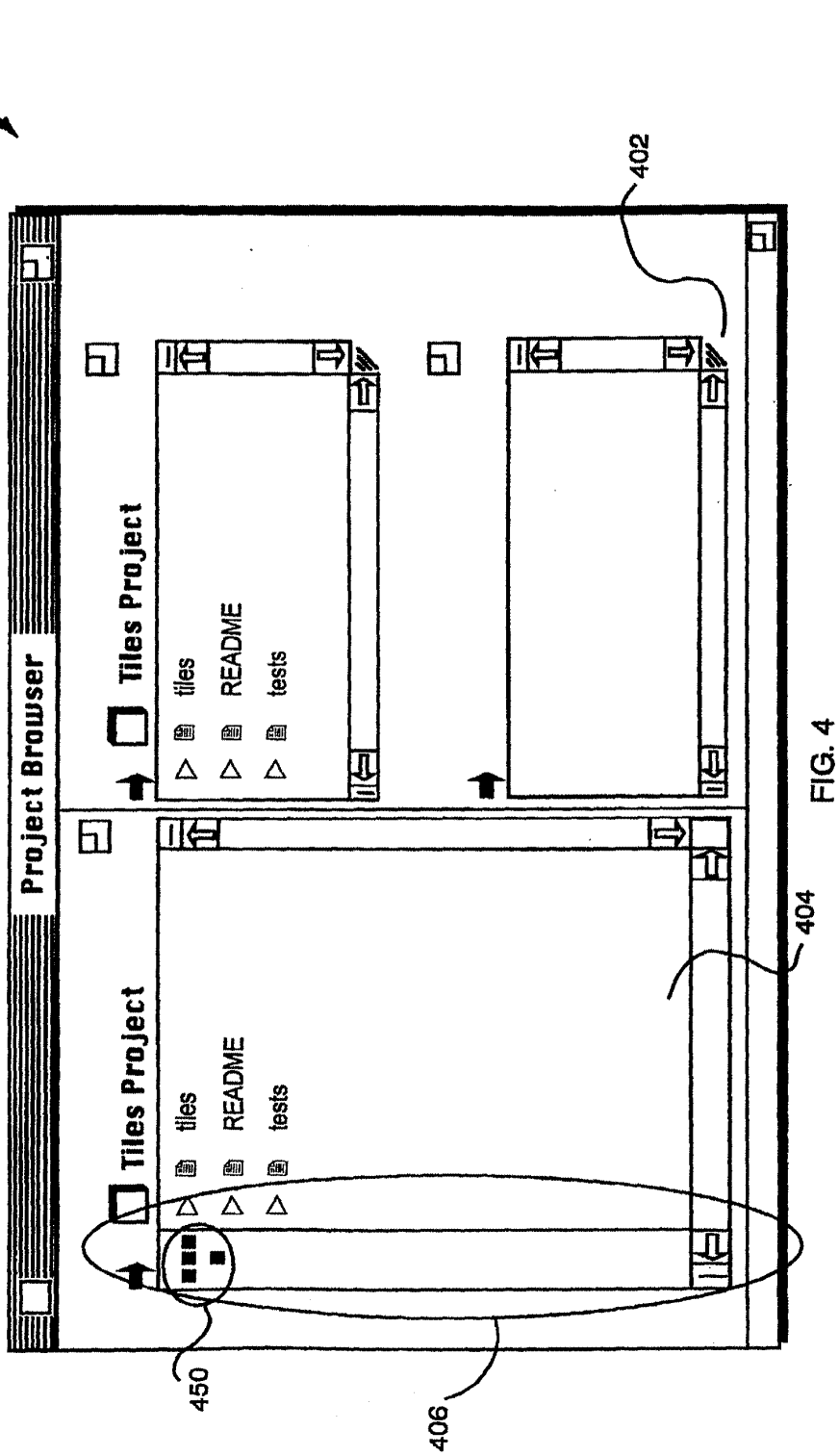


FIG. 4

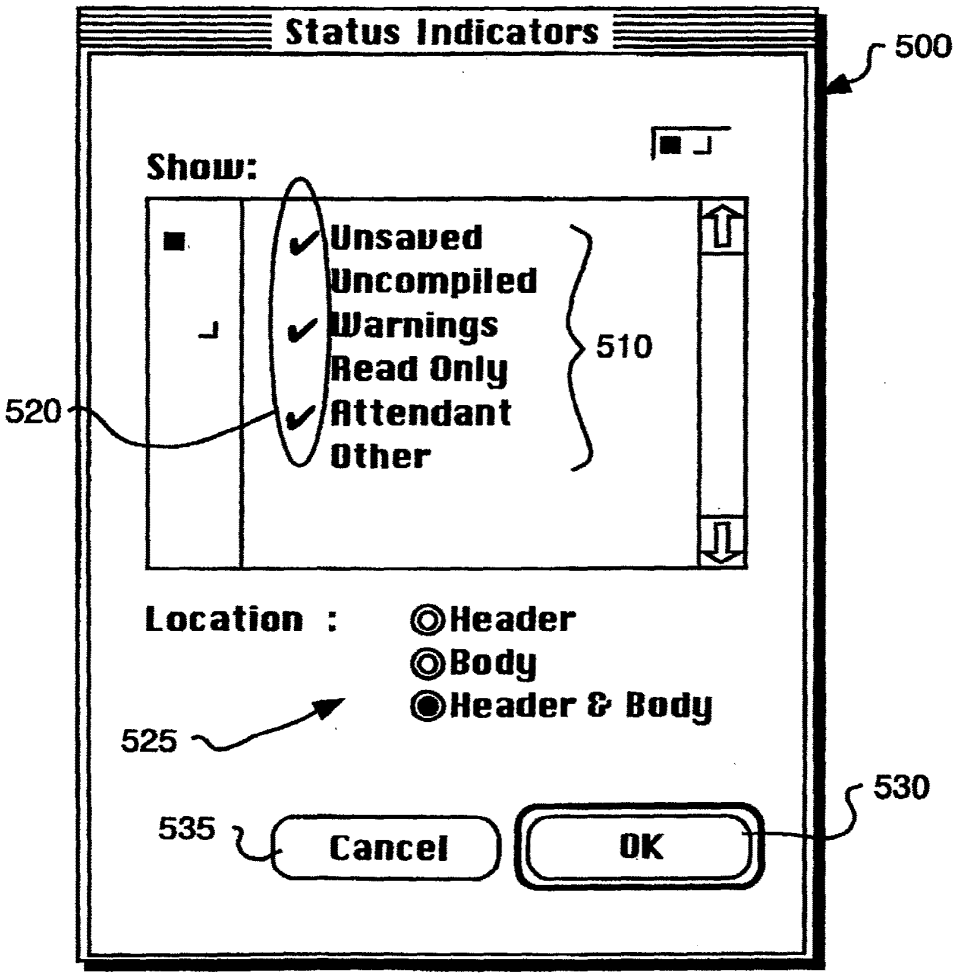

Status Indicators
500
Show:
Unsaved
Uncompiled
Warnings
Read Only
Attendant
Other
510
520
Location :
Header
Body
Header & Body
525
530
535
Cancel
OK

FIG. 5

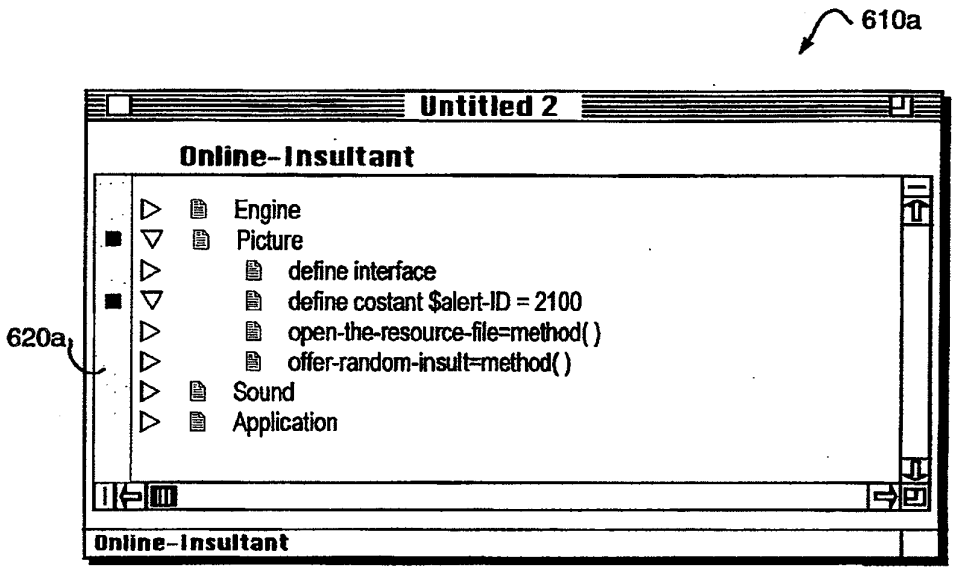


FIG. 6A

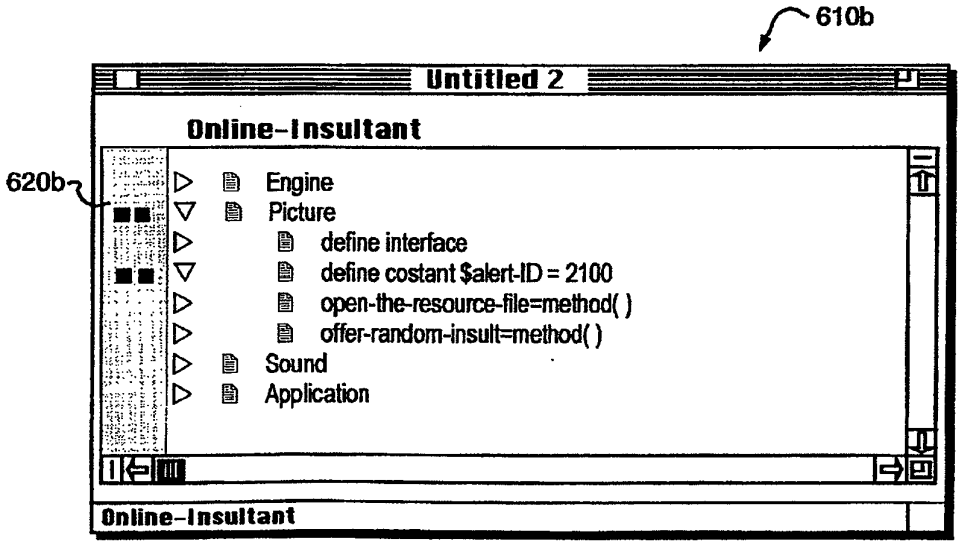


FIG. 6B

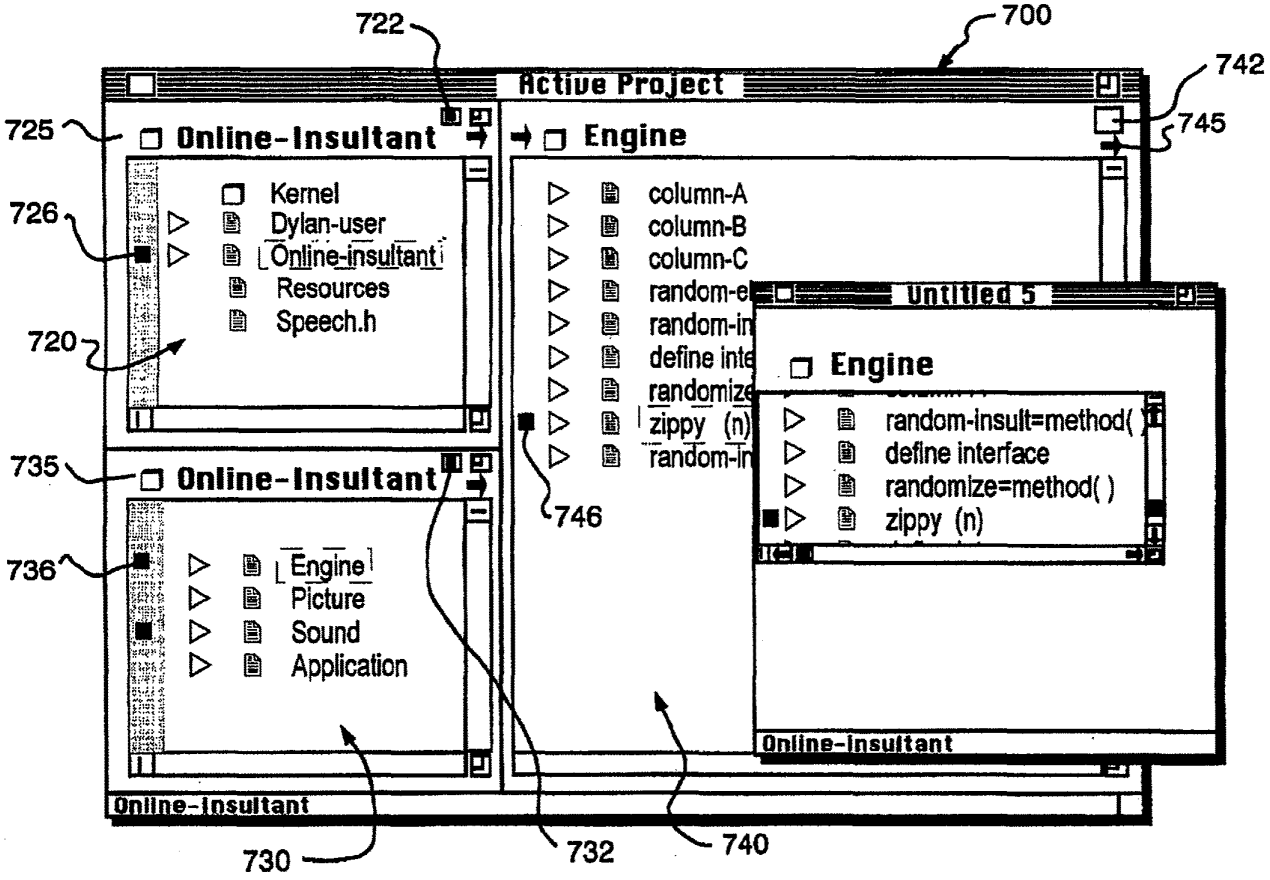

722
700
742
745
Active Project
725
Online-Insultant
Kernel
Dylan-user
Online-insultant
Resources
Speech.h
726
720
Engine
column-A
column-B
column-C
random-e
random-in
define inte
randomize
zippy (n)
random-in
746
Untitled 5
Engine
random-insult=method( )
define interface
randomize=method( )
zippy (n)
735
Online-Insultant
736
Engine
Picture
Sound
Application
Online-insultant
Online-insultant
730
732
740

FIG. 7

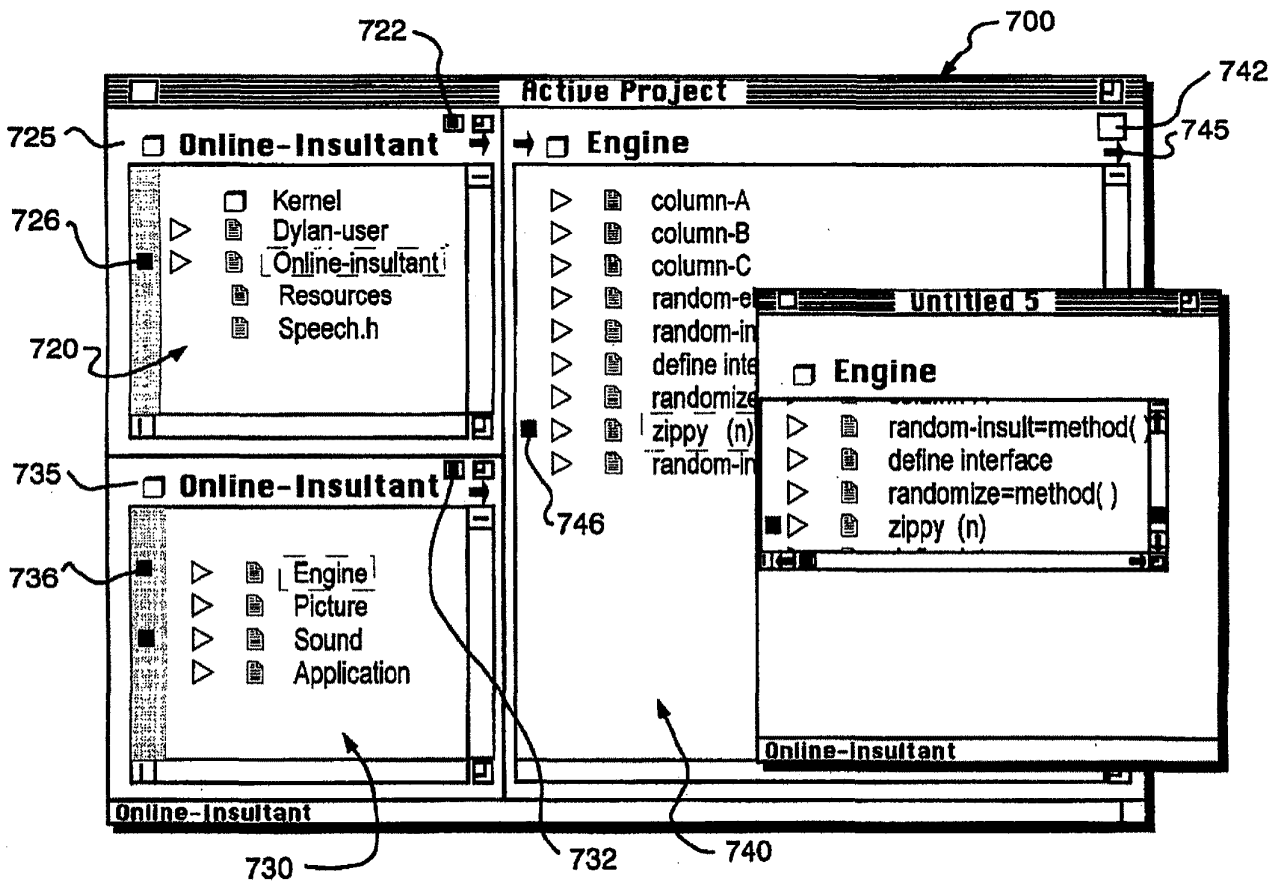


FIG. 7

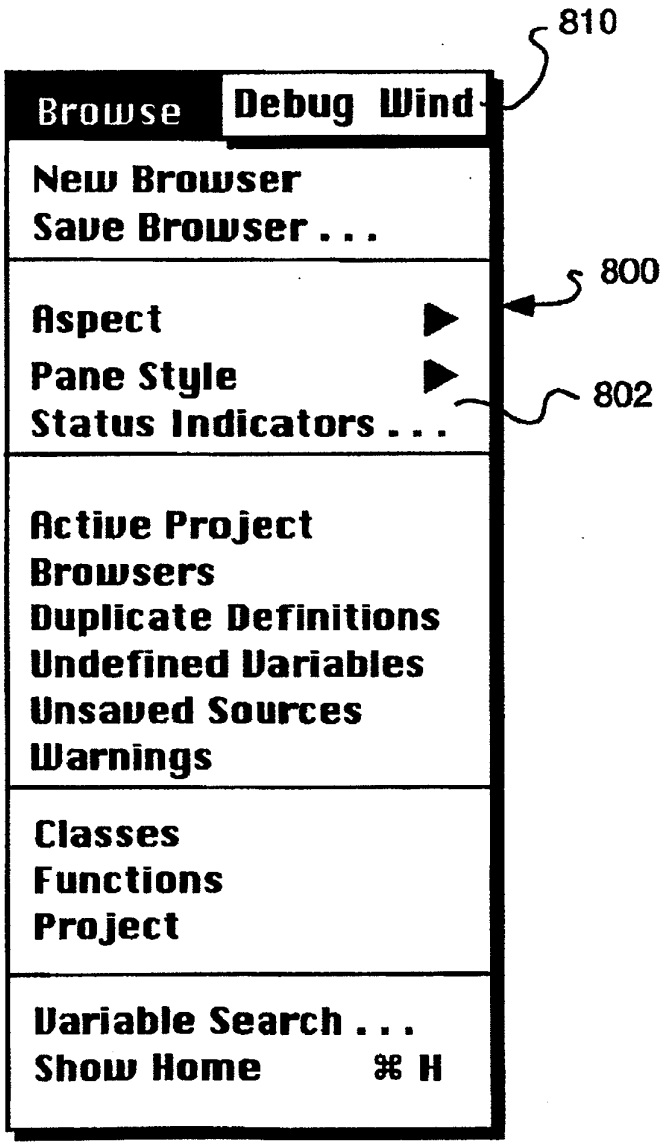


FIG. 8

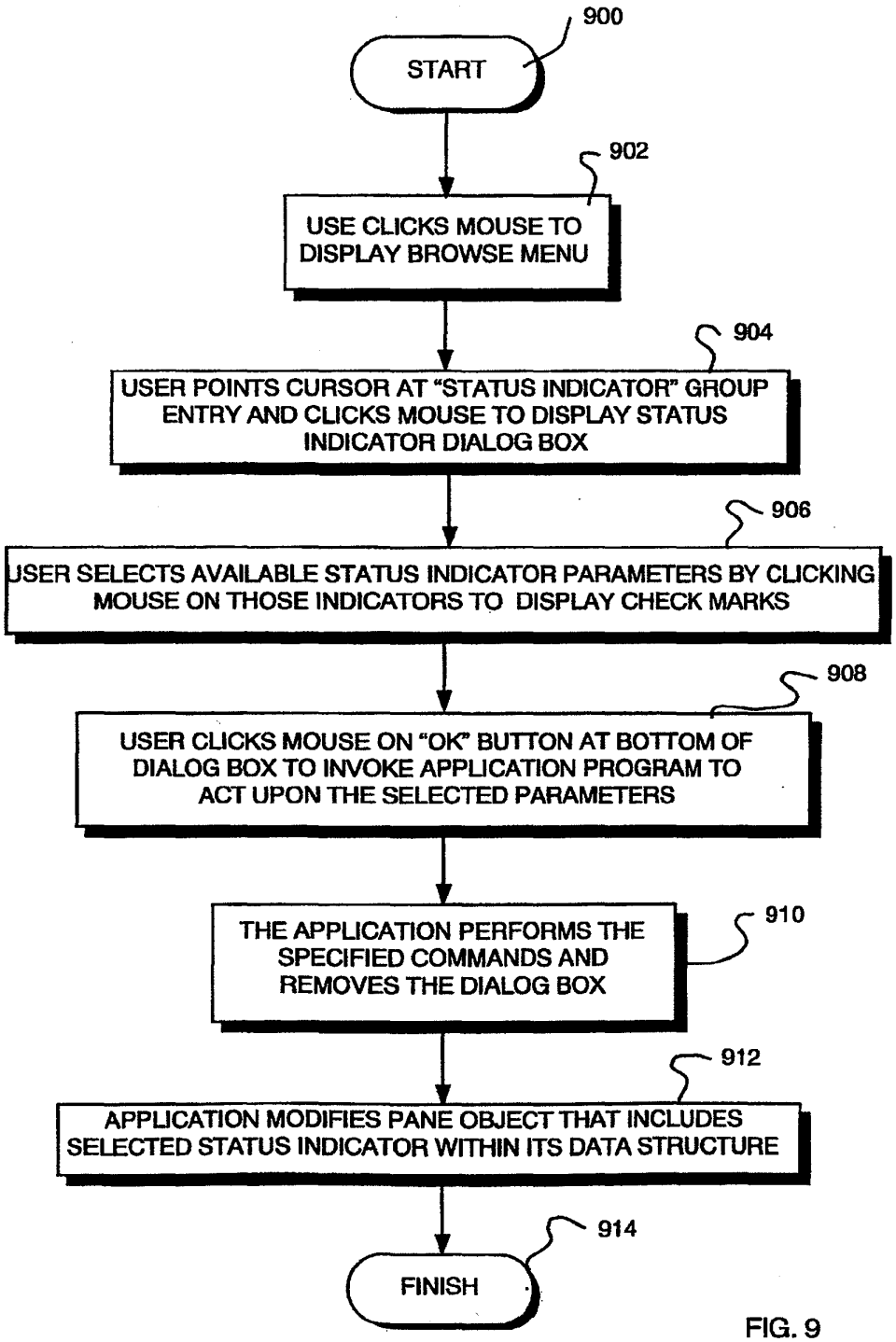


FIG. 9

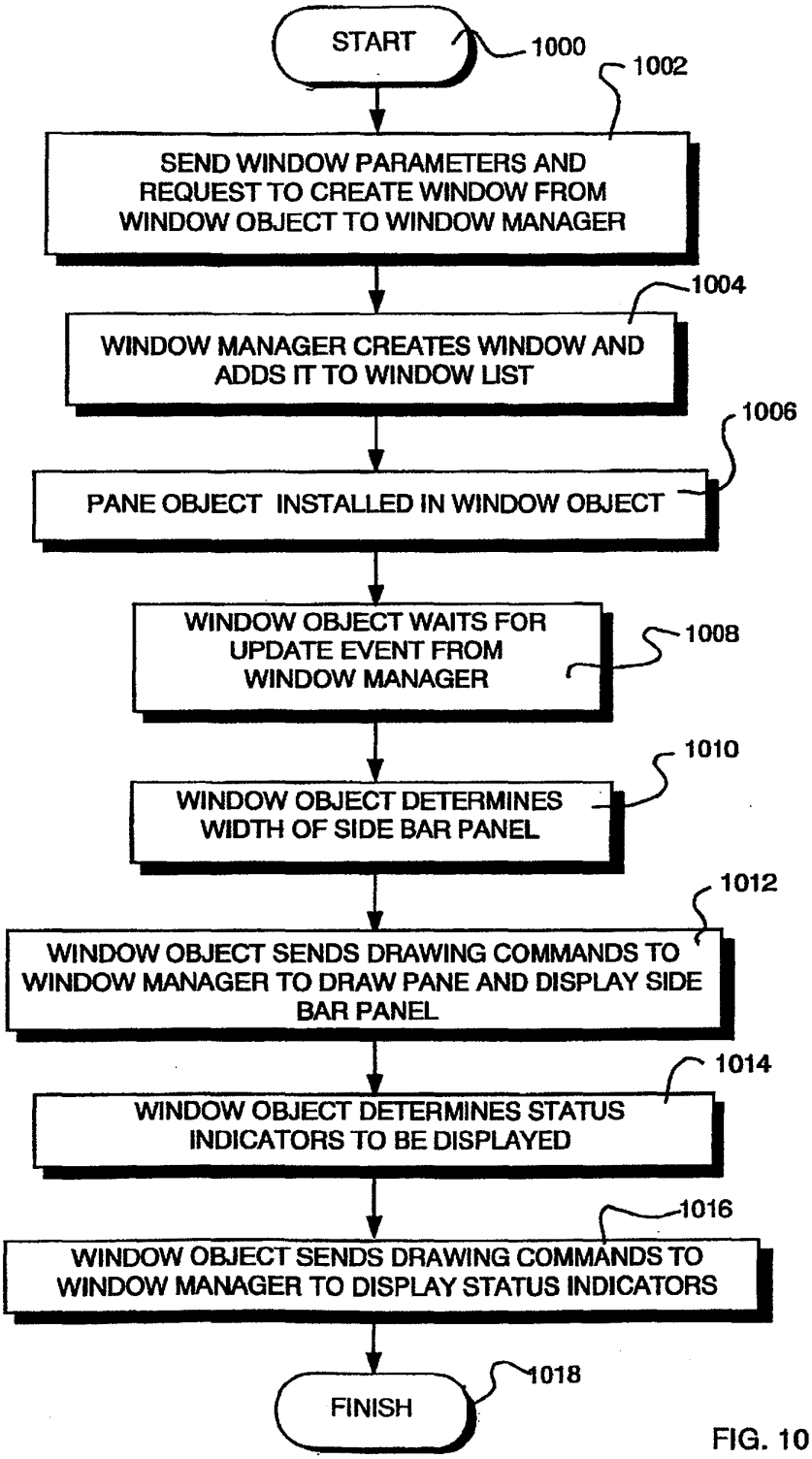


FIG. 10

STATUS INDICATORS OF AN IMPROVED GRAPHICAL USER INTERFACE

This is a continuation of U.S. patent application Ser. No. 08/245,877, filed on May 19, 1994, now abandoned.

CROSS-REFERENCE TO RELATED APPLICATIONS

This invention is related to U.S. patent application Ser. No. 08/246,319, titled ASPECT AND STYLE ELEMENTS OF AN IMPROVED GRAPHICAL USER INTERFACE, filed on May 19, 1994 and assigned to the assignee of the present invention. The invention is also related to U.S. patent application Ser. No. 08/050,510, titled INTERACTIVE USER INTERFACE, filed on Apr. 20, 1993 and assigned to the assignee of the present invention, which application is hereby incorporated by reference as though fully set forth herein.

FIELD OF THE INVENTION

This invention relates to interactive user interfaces for computer systems and, more specifically, to status indicators of a customizable browser framework for an improved graphical user interface.

BACKGROUND OF THE INVENTION

Graphical user interfaces are typically based on bit-mapped graphic display technology that employs iconic (pictorial) representations, multi-font typographic-style text and other art work on a display screen of a computer system. These interfaces include a window environment that configures the screen to resemble a graphical display for a user to enter or view information. Generally, an application program executing on the computer system presents the information to the user through a window by drawing images, graphics or text within the window region. The user, in turn, communicates with the application by "pointing" at objects in the window with a cursor that is controlled by a hand-operated pointing device, such as a mouse.

Transient "pop-up" or "pull-down" menus list command selections that are available to the user at all times, regardless of what else may be displayed on the screen. For example, there is no need to close a document and return to a main menu screen in order to select and issue commands from a menu. The menus can be activated and commands selected merely by pointing to them with the mouse. That is, the commands may be issued by actuating the mouse to move the cursor onto or near the command selection and pressing and releasing, i.e., "clicking", a button switch on the mouse.

In general, these menus may be manifested as direct access menus which show all possible choices on a window, such as with a panel of buttons, or as taxonomic menus which classify a domain hierarchy and allow the user to navigate through it. In many situations it is not necessary for the menu to remain permanently visible on the display screen and it can thus be "popped-up" on the screen when required. Typically, a button switch on the mouse is depressed to display the menu which is then painted on the screen near the cursor position. When the switch is released, the menu disappears. Again, menu selection is achieved by pointing the cursor at the desired item with the mouse, which visually shades the selected item.

The pop up menu is a convenient way to keep frequently used commands and information accessible without occupying space on the window. Alternately, several different menus can be provided by buttons on the window which, when selected by the mouse, display their menus; these are called pull-down menus. Typically, these menus remain drawn only while the mouse button is depressed.

Commands in the pull-down and pop-up menus typically act one or more objects. If an application requires more information to carry out a command, a dialog box may be employed. A dialog box is a special type of window that elicits information from the user, such as requiring the user to check items or fill in blanks.

Graphical user interfaces are exemplified by the Finder application software system used in the Macintosh® series of computers and manufactured by Apple Computer, Inc. of Cupertino, Calif. An aspect of that software system is described in U.S. Pat. No. 4,931,783, which is hereby incorporated by reference as though fully set forth herein. That patent describes how to manipulate and control the functions available from the Finder system though the use of menus. The details of how to implement such a system, as well as other Finder functions, are described in a volume of the publication "Inside Macintosh", published by Addison-Wesley, titled "Macintosh Toolbox Essentials", which volume is also hereby incorporated by reference.

Another example of a popular software system that provides a graphical user interface is the Windows® operating system, which is commercially available from Microsoft Corporation. The present invention is applicable to all such systems and is primarily concerned with accessing information relating to the states of objects developed in accordance with object-oriented programming techniques.

Object-oriented programming is a paradigm for designing and implementing software programs. Generally, object-oriented programming defines and packages objects, where an object consists of a data structure together with the operations available for that structure. Once such objects have been defined, it is possible to build a program as a simple sequence of processes to be performed on specified instances of these objects. An integral part of object definition is the ability to create new, more elaborate objects as enhancements of those previously defined.

When developing software programs, there may be attributes associated with one or more of these objects that a user, such as a programmer, would like to examine. These attributes may relate to numerical values or information concerning the status of an object, such as whether its source code has been modified. Typically, the attributes are represented as bits of state associated with the object and, in many conventional graphical user interface systems, these state attributes are available from menus accessible from a window. However, the programmer retrieves the menu and accesses each attribute with a control device, such as a mouse, to examine its contents which, typically, are manifested textually. Such procedures may be time consuming and inefficient, particularly if there are many different state attributes associated with the object.

Accordingly, it is among the objects of the present invention to provide an improved graphical user interface having the capability of positionally representing state attributes associated with an object.

Another object of the present invention is to provide an interactive graphical user interface having the capability of positionally displaying state attributes for more than one object at a time.

SUMMARY OF THE INVENTION

Briefly, an improved graphical user interface in accordance with the invention comprises novel status indicators

pertaining to state attributes associated with objects displayed on a display screen of a computer system. These status indicators are preferably portrayed on a portion of a window, called a pane, of a display screen as distinct visual cues that are located adjacent to their associated objects, thereby providing a customizable browser framework to a user, such as a programmer. Examples of object-related attributes represented by the status indicators include, among others, whether the source code of an object has been saved or compiled, and an indication of its user access/protection status, e.g., read-only.

Specifically, a bar panel is provided along a predetermined side (i.e., right/left, top or bottom) of each window pane for displaying the status indicators. According to one aspect of the invention, the status indicators may be distinguished from among one other by color, shape and/or relative spatial position along this side bar panel. In an illustrative embodiment, each type of indicator has a different color and a different location along the width of the panel, while assuming a rectangular, "blip-like" shape that is common among all indicator types.

In accordance with another aspect of the invention, the width of the side bar panel dynamically expands/contracts depending upon the quantity of status indicators chosen for display. The dynamically adjustable nature of the side bar panel allows numerous status indicators to be shown on the panel, if desired, while also conserving screen real estate when only a few indicators are selected for display.

Moreover, use of positional status indicators, each having a distinct color, shape and/or spatial position within the side bar panel of a pane, obviates the need for a user to read text associated with each indicator. This is particularly significant for a programmer that may be interested in the status of various objects during software development in a programming environment. By eliminating the need to read text to determine the status of these objects, the programmer can merely glance at the visual cues and off-load interpretation of their states to the subconscience, thereby eliminating time consuming and inefficient computer-related commands.

Operationally, the types of indicators portrayed on the screen for an object are preferably determined on a pane-by-pane basis by retrieving a status indicator dialog box for that pane from a pull-down menu or through other means such as double clicking the mouse in the side bar region of a pane. The dialog box is a user interface element that provides a list of available status indicator entries, including those that are currently being displayed; these latter indicators are so designated by check marks or other inscription adjacent to the entries. A user may select or deselect any of the available status indicators on the list for a pane by, for example, "clicking" on the indicator entries with a mouse.

The selected status indicators are stored in a data structure located within a pane object. These contents of the data structure determine which status indicators will be displayed in the pane which, in turn, determine the width of the dynamically adjustable side bar panel used to accommodate the indicators. These contents are also used to determine the appropriate commands to send to the window manager, which then displays the pane.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings, in which:

FIG. **1** is a block diagram of a computer system, such as a personal computer system, on which the invention may advantageously operate;

FIG. **2** is a block diagram showing the relationship of an operating system, an application program, a screen buffer and a display screen of the computer system of FIG. **1**;

FIG. **3** shows the interaction between an application program and a window manager in accordance with the invention;

FIG. **4** shows a display screen illustrating novel status indicators in accordance with the invention;

FIG. **5** shows a status indicator dialog box for displaying a list of available status indicator entries in accordance with the invention;

FIGS. **6**A and **6**B show window panes, each illustrating the dynamically-adjustable side bar panel in accordance with the invention;

FIG. **7** shows various panes of a window, each of which depicts a level of a containment hierarchy that may advantageously utilize the status indicators of the present invention;

FIG. **8** shows a status indicator group entry on a browse menu in accordance with the invention;

FIG. **9** is an illustrative flow-chart of the sequence of steps involved in operation of the improved graphical user interface when selecting the status indicators of the present invention; and

FIG. **10** is an illustrative flow-chart of the sequence of steps used to display a window and associated panes, the panes containing status indicators in accordance with the invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

FIG. **1** illustrates a computer system **100** comprising a central processing unit (CPU) **110** coupled between a memory **114** and input/output (I/O) circuitry **118** by bidirectional buses **112** and **116**. The memory **114** typically comprises random access memory (RAM) for temporary storage of information, including an application program (not shown), and read only memory (ROM) for permanent storage of the computer's configuration and basic operating commands, such as portions of an operating system (not shown). As described further herein, the application program and operating system interact to control the operations of the CPU **110** and computer system **100**.

The I/O circuitry **118** is, in turn, connected to a mass storage unit **120**, such as a disk drive, via a bidirectional bus **122** and to cursor control devices, such as a keyboard **124** (via cable **126**) and a mouse **130** (via cable **128**). A conventional display monitor **132** having a display screen **135** is also connected to I/O circuitry **118** via a cable **138**. Specifically, the I/O circuitry **118** receives information, such as control and data signals, from the mouse **130** and keyboard **124**, and provides that information to the CPU **110** for storage on the mass storage unit **120** or for display on the screen **135**. It is to be understood that the I/O circuitry contains the necessary hardware, e.g., buffers and adapters, needed to interface with the mouse, keyboard and display monitor.

The mouse **130** typically contains at least one button switch **134** operated by a user of the system. A cursor **140** is displayed on the screen **135** and its position is controllable via the mouse **130** or the keyboard **124**, as is well known. An example of the mouse **130** is shown and described in U.S. Pat. No. Re. 32,632, which patent is hereby incorporated by reference as though fully set forth herein.

The computer system **100** is preferably a personal computer of the Macintosh® series of computers sold by Apple

Computer, Inc., although the invention may also be practiced in the context of any computer. These computers have resident thereon, and are controlled and coordinated by, operating system software, such as the Apple® System/7® operating system.

A window environment is typically displayed on the screen **135**. In accordance with an aspect of the invention described herein, the window environment includes windows **142**, each of which may contain panes **144**, with each pane covering a portion of a window **142**. The window environment is generally part of the operating system software that includes a collection of utility programs for controlling the operation of the computer system **100**. The operating system, in turn, interacts with an application program to provide higher level functionality, including a direct interface with the user. Specifically, the application program makes use of operating system functions by issuing a series of task commands to the operating system which then performs the requested task. For example, the application program may request that the operating system display certain information on the panes **144** of the display screen **135** for presentation to the user.

The invention herein features, along with these windows and panes, the provision of new user interface elements, such as menus and their entries, which, when invoked, cause actions to take place that enhance the ability of a user to interact with the computer system. This new behavior of the system is brought about by the interaction of these elements with a series of system software routines associated with the operating system. These system software routines, in turn, interact with the application program to create the windows and panes, and manage the new menus, as described further herein.

FIG. **2** is a schematic illustration of the interaction of an application program **202** and an operating system **204** of a computer system **200**, which is similar to, and has equivalent elements of, the computer system **100** of FIG. **1**. The application program **202** and the operating system **204** interact to control and coordinate the operations of the computer **200** and their interaction is illustrated schematically by arrow **206**. In order to display information on a screen display **235**, application program **202** generates and sends display commands to a window manager program **205** of the operating system **204**. The window manager program **205** stores the information directly (via arrow **208**) into a screen buffer **210**.

The window manager **205** is a system software routine that is generally responsible for managing the windows that the user views during operation of the application program; the application program is generally responsible for managing the panes. That is, it is generally the task of the window manager to keep track of the location and size of the window and window areas which must be drawn and redrawn in connection with elements of the improved graphical user interface of the present invention. Further details relating to the window manager are provided in the aforementioned "Inside Macintosh" publication "Macintosh Toolbox Essentials".

Under control of various hardware and software in the system, the contents of the screen buffer **210** are read out of the buffer and provided, as indicated schematically by arrow **212**, to a display adapter **214**. The display adapter contains hardware and software (sometimes in the form of firmware) which converts the information in the screen buffer **210** to a form which can be used to drive a display screen **235** of a monitor **232** which is connected to display adapter by cable **218**.

In a preferred embodiment, the invention described herein is implemented in an object-oriented dynamic programming language, such as Common Lisp, using object-oriented programming (OOP) techniques. The Common Lisp language is well-known and many articles and texts are available which describe the language in detail. In addition, Common Lisp compilers are available from several vendors. Accordingly, for reasons of clarity, the details of the Common Lisp language and the operation of the Common Lisp compiler will not be discussed further in detail herein.

As will be understood by those skilled in the art, OOP techniques involve the definition, creation, use and destruction of "objects". These objects are software entities comprising data elements and routines, or functions, which manipulate the data elements. The data and related functions are treated by the software as an entity that can be created, used and deleted as if it were a single item. Together, the data and functions enable objects to model virtually any real world entity in terms of its characteristics, which can be represented by the data elements, and its behavior, which can be represented by its data manipulation functions. In this way, objects can model concrete things like people and computers, while also modeling abstract concepts like numbers or geometrical designs.

Objects are defined by created "classes" which act as templets that instruct the compiler how to construct an actual object. A class may, for example, specify the number and type of data variables and the steps involved in the functions which manipulate the data.

The principle benefits of OOP techniques arise out of three basic principles: encapsulation, polymorphism and inheritance. More specifically, objects can be designed to hide, or encapsulate, all, or a portion of, its internal data structure and internal functions. Polymorphism is a concept which allows objects and functions which have the same overall format, but which work with different data, to function differently in order to produce consistent results. Inheritance on the other hand, allows program developers to easily reuse pre-existing programs and to avoid creating software from scratch. The principle of inheritance allows a software developer to declare classes (and the objects which are later created from them) as related. Specifically, classes may be designated as subclasses of other base classes. The creation of a new subclass which has some of the functionality (with selective modification) of another class allows software developers to easily customize existing code to meet their particular needs.

In accordance with the invention, the window and its panes are "objects" created by the application program to communicate with the window manager, which is preferably an object-oriented program. The interaction between an application program and a window manager is illustrated in greater detail in FIG. **3**.

In general, an application program **308** interfaces with the window manager **310** by creating and manipulating objects. The window manager itself is an object which is created when the operating system is started. Specifically, the application program creates window objects **300** that cause the window manager to create associated windows on the display screen. This is shown schematically by an arrow **302**. In addition, the application program **308** creates individual pane objects **350** that are stored in each window object **300**, as shown schematically by arrows **352** and **354**.

Since many pane objects may be created in order to display many panes on the display screen, the window object **300** communicates with the window manager **310** by means

of a sequence of drawing commands issued from the window object **300** to the window manager **310**, as illustrated by arrow **304**. The application **308** also communicates with the window manager **310** by sending commands to the manager **310**, as indicated by arrow **316**. The window manager **310** maintains a window list **312** that contains each window currently in the system.

Although object-oriented programming offers significant improvements over other programming concepts, program development still requires significant outlays of time and effort, especially if no preexisting software programs are available for modification. Consequently, a prior art approach has been to provide a program developer with a set of pre-defined, interconnected classes which create a set of objects and additional miscellaneous routines that are all directed to performing commonly-encountered tasks in a particular environment. Such predefined classes and libraries are typically called "application frameworks" and essentially provide a pre-fabricated structure for a working application.

For example, an application framework for a conventional graphical user interface might provide a set of pre-defined graphic interface objects which create windows, scroll bars, menus, etc. and provide the support and "default" behavior for these graphic interface objects. Since application frameworks are based on object-oriented techniques, the predefined classes can be used as base classes and the built-in default behavior can be inherited by developer-defined subclasses and either modified or overridden to allow developers to extend the framework and create customized solutions in a particular area of expertise. This object-oriented approach provides a major advantage over traditional programming since the programmer is not changing the original program, but rather extending the capabilities of the original program. In addition, developers are not blindly working through layers of code because the framework provides architectural guidance and modeling and, at the same time, frees the developers to supply specific actions unique to the problem domain.

There are many kinds of application frameworks available, depending upon the level the system involved and the kind of problem to be solved. The types of frameworks range from high-level application frameworks that assist in developing a user interface, to lower-level frameworks that provide basic system software services such as communications, printing, file system support, graphics, etc. Commercial examples of application frameworks include MacApp (Apple), OWL (Borland), NeXT Step App Kit (NEXT) and Smalltalk-80 MVC (ParcPlace).

While the application framework approach utilizes all the principles of encapsulation, polymorphism and inheritance in the object layer, and is a substantial improvement over other programming techniques, there are difficulties which arise. These difficulties are caused by the fact that it is easy for developers to reuse their own objects, but it is difficult for the developers to use objects generated by other programs. Further, application frameworks generally consist of one or more object "layers" on top of a monolithic operating system and even with the flexibility of the object layer, it is still often necessary to directly interact with the underlying operating system by means of awkward procedure calls.

In the same way that an application framework provides the developer with prefab functionality for an application program, a system flamework, such as that included in the preferred embodiment, can provide a prefab functionality for system level services which developers can modify or override to create customized solutions, thereby avoiding the awkward procedure calls necessary with the prior art application flameworks programs. For example, consider a customizable browser flamework which can provide the foundation for browsing and direct manipulation of objects. An application program developer who needed these capabilities would ordinarily have to write specific routines to provide them. To do this with a framework, the programmer only needs to supply the characteristics and behavior to the finished output, while the flamework provides the actual routines which perform the tasks.

A preferred embodiment takes the concept of frameworks and applies it throughout the entire system, including the application and operating system. For the commercial or corporate developer or systems integrator, this means all of the advantages that have been illustrated for a flamework, such as MacApp, can be leveraged not only at the system level for such services as printing, graphics, multi-media, file systems and I/O operations, but also at the application level, for things such as text and, as described herein, graphical user interfaces.

Referring again to FIG. **3**, the window object **300** and the pane object **350** are elements an improved graphical user interface having a customizable browser framework for greatly enhancing the ability of a user to navigate or browse through many different objects stored in the memory **114**. Specifically, as described further herein, the customizable browser framework classifies and organizes these latter objects as source code entities according to the semantics of the programming language. This type of browser is a particularly useful tool for the programmer that is developing software in a programming environment.

However, during such development, there may be state attributes associated with objects that the programmer would like to examine. These attributes may include whether the source code of an object has been saved or compiled, and the user access status of the code. In accordance with the present invention, the improved graphical user interface includes status indicators relating to these state attributes.

FIG. **4** shows a display screen **400** illustrating the novel status indicators, collectively referred to at **450**, which are associated with various objects of a project "Tiles". As can be seen, the indicators are displayed on a pane **404** of a window **402** as distinct visual cues adjacent to their associated objects; specifically, there are three (3) status indicators positioned alongside of the object "tiles" and one (i) indicator located next to the object "README". A novel side panel bar **406** is provided within the pane **404** for displaying these indicators.

According to one aspect of the invention, the status indicators **450** are distinguished from among one other by color, shape and/or relative spatial position along the side bar panel **406**. Preferably, each type of indicator described herein has a different color and a different relative location along the width of the panel; however, in the illustrative embodiment, each also has a common rectangular, "blip-like", i.e., simple geometric, shape.

FIG. **5** shows a status indicator dialog box **500** displaying a list of status indicator parameter entries **510** in connection with the present invention. As noted, these entries relate to the status of an object and include such state attributes as: (i) UNSAVED, (ii) UNCOMPILED, (iii) WARNINGS, (iv) READ ONLY, (v) ATTENDANT and (vi) OTHER. Specifically, the object-related attribute represented by the entry UNSAVED indicates that the source code of the object has not been saved since modification; the entry UNCOM-

PILED indicates that the object's source code has not yet been compiled; the entry WARNINGS indicates that errors have been detected in the source code; the entry READ-ONLY indicates that the user has only read access (e.g., not write access) to the object's code; and the entry ATTEN-DANT indicates that object's source code is not designated as a runtime object for execution by the computer.

The entry OTHER is a special entry that is only active when one or more status indicators are not displayed. The status indicators that are currently displayed, however, are designated by check marks **520** or other similar inscription adjacent to the indicators. A user may select or deselect any of the available status indicators on the list for display on the screen by, e.g., "clicking" on the indicator entries with a mouse, which toggles the entry's state.

The dialog box **500** is preferably a modal dialog box because it puts the user in "mode" of being able to work only inside the box **500**. That is, whether accepting/rejecting the selected/deselected status indicator entries **510**, the user must affirm that choice by clicking on either the "OK" button **530**, located at the bottom of the box, or the adjacent "Cancel" button **535**, prior to issuing any other commands to the application.

In accordance with the invention, the status indicators may appear along any side of a frame of the pane; that is, the indicators may appear in a side bar located on the right or left side of the pane, or they may appear in a title bar arranged along the top side of each pane or, if appropriate, in a bar located along the bottom side of the pane. Preferably, a location pop-up **525** specifies whether these indicators should appear in the right/left side bar, the title bar, in both areas or in neither area. When they appear in the title bar, the status indicators apply to the object that is being displayed in the pane.

The text for each indicator (i)–(vi) is preferably displayed in the color of that indicator. In the illustrative embodiment, the preferred color of the positional cue (and its associated text) for each status indicator is as follows:

(i) UNSAVED	blue
(ii) UNCOMPILED	green
(iii) WARNINGS	orange
(iv) READ ONLY	pink
(v) ATTENDANT	yellow
(vi) OTHER	grey

The set of status indicators shown is determined on a pane-bypane basis. Accordingly, and referring again to FIG. **3**, each pane object **350** created by the application program **308** preferably has an internal data structure **356** containing a list of every status indicator displayed by the dialog box **500**, together with the status of each indicator, i.e., selected or unselected. As described in connection with FIG. **5**, the status of each indicator is determined by a user manipulating the dialog box **500**. In addition, each window object **300** contains a list of panes **306** and each pane object **350** also contains a list of objects. The state of each object displayed in the pane determines whether or not to draw each possible status indicator.

In accordance with another aspect of the invention, the width of the novel side bar panel **406** (FIG. **4**) dynamically expands/contracts depending upon the quantity of status indicators chosen for display. FIGS. **6**A and **6**B show dynamically-adjustable side bar panels **620***a,b* of panes **610***a,b*. The dynamically adjustable nature of the side panel **620** allows numerous status indicators to be shown on the panel, if desired, while also conserving screen real estate when only a few indicators are selected for display.

Specifically, each indicator (i)–(vi) appearing in a right/left side bar panel preferably has a relative spacial location along that bar panel that is determined on a pane-by-pane basis by the user depending upon the quantity of indicators chosen for display. For example, if status indicators (i), (ii) and (vi) are selected for display in a particular pane, the width of the side bar panel would consist of three (3) columns, with each column being designated a status indicator. If, on the other hand, the indicators appear in a title (or bottom) bar panel of the pane, each indicator would occupy an analogous spatial "row" of that particular side bar panel that would be determined, again, on a pane-by-pane basis depending upon the number of status indicators chosen for display. That is, the relative spatial positions of indicators are consistent within a pane and unoccupied columns (or rows) are eliminated. It will be apparent to those skilled in the art, however, that other predetermined spacial position configurations of these indicators, such as a diagonal configuration or a fixed-location configuration, may be employed within the teachings of the invention.

As an example of the latter configuration, each indicator (i)–(vi) would have a fixed spacial location along the side bar, with the indicator (i) occupying the first spacial "column" nearest to the border of the pane and the indicator (vi) occupying the last spatial column farthest from that border. If the indicators appear in a title (or bottom) bar panel of the pane, each indicator would occupy an analogous spacial "row" of that particular side bar panel. In accordance with this embodiment, if an indicator (vi) is displayed for any object present in the pane, then the side bar panel will be drawn to its maximum width; however, if only the indicator (i) is displayed for each object in the pane, the width of side bar panel **620** will be minimal.

Use of the status indicators of the present invention may be particularly advantageous with respect to the customizable browser framework described herein when used in a programming development environment having a containment hierarchy for developing programs. FIG. **7** shows various panes **720**–**740** of a window **700**, with each pane depicting a level of the containment hierarchy that advantageously utilizes the status indicators of the present invention. Specifically, the containment hierarchy allows source code manipulations and operations, such as "saving" of the code, within the various levels of hierarchy. Here, objects, e.g., functions or class definitions, are units of program source code called "source entities" and comprise the lowest level of the hierarchy. A "source container", resident at the next level, contains an ordered collection of these entities. The source container is analogous to a file and each source entity resides in at least one source container. A module comprises at least one source container and is itself incorporated with an entire project. The project is the center of source code management and contains a working set of source entities.

In FIG. **7**, the pane **720** depicts the project level of the containment hierarchy, where various modules of the project "Online-Insultant" are displayed. A status indicator **722** associated with the project Online-Insultant in bar **725** indicates that the project level is the basis of the pane **720**. A module "Online-Insultant" is selected and has its contents displayed in pane **730** (the module level of the hierarchy). Status indicator **732** is associated with the module Online-Insultant in the bar **735**. Various source containers are displayed in this pane **730**, including a selected container "Engine".

Pane **740** shows this source container level "Engine" having a status indicator **742** in bar **745**. A selected source entity "zippy" is among the source entities listed in the pane **740**. (A pane **750**, similar to pane **740**, is shown overlapping the pane **740**.)

A feature of the present invention is that when a modification is made at, e.g., the source entity level of the project, the status indicators propagate up through the hierarchy to the project level, thereby notifying the programmer that each of those levels require modification. Accordingly, the status indicator **746** (preferably an UNSAVED indicator) is displayed for the source entity "zippy" and indicates that its source code has not been saved. Because "zippy" is contained in source container "Engine", its status indicator **742** is displayed in pane **740**, as is status indicator **736** in pane **730**. Module "Online-Insultant", comprising source container "Engine", also has its status indicators **732** and **726** displayed in panes **730** and **720**, and, lastly, project "Online-Insultant" has its status indicator **722** displayed in pane **720**. Such "linking" between the panes **720–740** is made possible because of the contents of the data structure **306** contained in the window object **300** of FIG. **3**.

FIG. **8** shows a Status Indicator group entry **802** that is displayed on a Browse menu **800**. The Browse menu is preferably a pull-down menu that is available from a menu bar **810**. The "Status Indicator" entry **802** is active whenever there is an active pane in the window **820**.

FIGS. **1–8** will now be referenced in connection with sequence of steps illustrated in the flow-chart of FIG. **9** to describe the operation of the improved graphical user interface when selecting the status indicators.

To establish (or change) the status indicators displayed in the side bar panel **620** of an active window pane, the sequence starts in Step **900** and proceeds to Step **902** where a user, operating the mouse **130** to control the cursor **140** on display screen **135**, "points" the cursor at the Browse menu **800** of menu bar **810**. The user then "clicks" the switch **134** of the mouse to display that menu **800**.

In Step **904**, the user manipulates the mouse so that the cursor points to the "Status Indicator" group entry **802** on menu **800** and clicks on that entry so as to display the status indicator dialog box **500**. As noted, the status indicators portrayed for a particular object are preferably determined on a pane-by-pane basis; accordingly, the selection of indicators made by the user will determine which indicators are displayed in the currently active pane.

In Step **906**, the user then selects (or deselects) any of the available status indicators on the list by clicking on those indicator parameter entries. The status indicator parameters selected for display are preferably noted by the presence of check marks **520**. The user then clicks the "OK" button **530** at the bottom of the dialog box **500** in Step **908** to invoke the application program **308** to act upon the selected parameters. In response, in Step **910**, the application **308** performs the specified command provided by the user and removes the box **500** from the screen. If, however, the user selects the "Cancel" button **535**, the application discards any changes issued by the user since the modal dialog box was displayed and then removes the box. The application **308** then modifies a pane object **350** that includes the selected status indicator parameters within the contents of its data structure **356**, as indicated in Step **912**. The routine then finishes in Step **914**.

As previously noted, a window object and a pane object interact with the window manager to provide various window management functions, such as creating a window and its panes. An illustrative routine used by the application program to display a window and its associated panes, the latter of which contain status indicators, is shown in the flowchart of FIG. **10**. The routine starts in Step **1000** and proceeds to Step **1002** in which the parameters defining the window, e.g., its size and interface elements, together with a request to create a new window, are sent (illustrated as arrow **304** in FIG. **3**) from the window object to the window manager. In response to this request, the window manager creates a window (Step **1004**) and adds the new window to the window list **310**. The list of created windows allows the window manager to keep track of the locations and sizes of the windows which must be drawn and redrawn as windows are created and deleted in connection with elements of the improved graphical user interface of the present invention.

In Step **1006**, a pane object **350** is installed in the window object **300** and in Step **1008**, the window object waits for an update event from the window manager indicating that the window and pane needs to be drawn. The window manager generates such update events to coordinate the display of the windows and panes. Upon receiving the update event, the window object, in Step **1010**, determines the width of the side bar panel using a count of status indicator parameters stored in the pane object's list **356** and, in Step **1012**, sends appropriate drawing commands to the window manager to draw the pane and display the side bar panel. In step **1014**, the window object **300** then determines which status indicators should be shown for each object stored in the data structure **356** of the pane object and, in Step **1016**, the window object sends appropriate drawing commands to the window manager to display those status indicators. The routine then finishes in Step **1018**.

An advantage of the present invention relates to the use of visual positional status indicators, each having a distinct color, shape and/or spatial position within the side bar panel, to obviate the need for a user to read text associated with each indicator. This is particularly significant for a programmer that may be interested in the status of various objects during software development in a programming environment. By eliminating the need to read text to determine the status of these objects, the programmer can merely glance at the visual cues and off-load interpretation of their states to the sub-conscience, thereby eliminating time consuming and inefficient computer-related commands.

While there has been shown and described an illustrative embodiment for manifesting status indicators on a window pane as distinct visual cues within a program development environment, it is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention. For example, the invention is also applicable to a relational database environment where users have a need to share records. In this embodiment of the invention, a user may scan through records of the database and retrieve a record on a screen for editing purposes. Prior to initiating the edit, the status indicators described herein may be configured to propagate onto all screens of users accessing the database to inform them of the record access. Here, the status indicators may be distinguished from among one another by shape, e.g., squares, triangles, circles, etc. In another embodiment, these status indicators may be employed in a networked development environment as a source code control mechanism to govern access to particular files.

The foregoing description has been directed to specific embodiments of this invention. It will be apparent, however, that other variations and modifications may be made to the described embodiments, with the attainment of some or all of their advantages. Therefore, it is the object of the

appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

What is claimed is:

1. An improved graphical user interface of a computer system having a display monitor for displaying a cursor on a display screen, said screen having associated therewith a device for controlling said cursor, said interface comprising:

a window of said screen, said window configured for apportionment into at least one pane for displaying objects of said system; and

status indicator means for positionally displaying state attributes associated with said objects, said status indicator means being displayed on a side bar panel of said pane, said side bar panel having a dynamically-adjustable width configured to one of expand and contract in response to the quantity of said state attributes selected for display.

2. The improved graphical user interface of claim 1 wherein said status indicator means comprises distinct visual cues.

3. The improved graphical user interface of claim 2 wherein said visual cues are distinguished from among one other by shape.

4. The improved graphical user interface of claim 2 wherein said visual cues are distinguished from among one other by color.

5. The improved graphical user interface of claim 4 further comprising means for setting said status indicator means to display selected ones of said visual cues.

6. The improved graphical user interface of claim 5 wherein said status indicator setting means comprises a status indicator dialog box element displayed on said display screen, the user manipulating said cursor controlled device to select one of a plurality of status indicator parameter entries displayed on said dialog box element.

7. The improved graphical user interface of claim 6 wherein said visual cues are distinguished from among one other by relative spacial position along said dynamically-adjustable side bar panel.

8. A computer system having an improved graphical user interface for enhancing the ability of a user to interact with said system, said computer system including a display monitor for displaying a cursor on a display screen having associated therewith a device for controlling said cursor, said computer system comprising:

a processor;

an operating system cooperating with said processor to control said display screen;

a window manager object created by said operating system, said window manager object drawing a pane for display on said display screen; an application program for generating source entity objects for display on said pane; and

user interface element means for positionally displaying state attributes associated with said source entity objects, said user interface means being displayed on a side bar panel of said pane, said side bar panel having a dynamically-adjustable width configured to one of expand and contract in response to the quantity of said state attributes selected for display.

9. The computer system of claim 8 wherein said user interface element means comprises status indicators having a blip-like shape and being distinguished from among other status indicators by color.

10. The computer system of claim 8 wherein said user interface element means comprises status indicators having a blip-like shape and being distinguished from among other status indicators by relative spacial position along said dynamically-adjustable side bar panel.

11. The computer system of claim 10 wherein said user interface element means further comprises a status indicator dialog box element for setting said status indicators in response to the user manipulating said cursor controlled device to select one of a plurality of status indicator parameter entries displayed on said dialog box element.

12. The computer system of claim 11 further comprising a window object created by said application program, said window Object communicating with said window manager object by issuing a sequence of drawing commands to display said pane.

13. The computer system of claim 12 further comprising a pane object created by said application program and stored in said window object, said pane object having an internal data structure containing a list of said selected status indicator parameter entries.

14. A method for enhancing the ability of a user to interact with a computer system having an improved graphical user interface, said computer including a display monitor for displaying a cursor on a display screen having associated therewith a device for controlling said cursor, said method comprising the steps of:

creating a window manager object from an operating system of said computer;

drawing a pane for display on said display screen using said window manager object;

generating source entity objects for display on said pane with an application program;

selecting status indicator parameters from a status indicator dialog box element of said improved graphical user interface displayed on said display screen in response to the user manipulating said cursor controlled device, said status indicator parameters positionally displaying state attributes associated with said source entity objects on a dynamically-adjustable side bar panel of said pane; and

one of expanding and contracting a width of said dynamicallyadjustable side bar panel in response to the quantity of said status indicator parameters selected for display.

15. The method of claim 14 further comprising the steps of:

creating a window object from said application program;

issuing a sequence of drawing commands from said window object to said window manager object; and

displaying said pane in response to said drawing commands.

16. The method of claim 15 further comprising the steps of:

creating a pane object from said application program;

storing said selected status indicator parameters in a list of an internal data structure of said pane object;

storing said pane object in said window object; and

sending said status indicator parameters to said window manager object when displaying said pane.

* * * * *

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/821,004	03/20/97	CHRISTENSEN S	04860.P1365C

E6M1/1030

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES CA 90025

EXAMINER	
DELA TORRE,C	
ART UNIT	**PAPER NUMBER**
2415	15

DATE MAILED: 10/30/97

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

PTO-90C (REV. 2/95)

08/821004

1 - File Copy