

EXHIBIT 1.04

does with the Button Bar and Menu Bar, but the results are the same.

- Click on the Options button, and the Status Bar Options dialog box appears (see fig. 26.44).

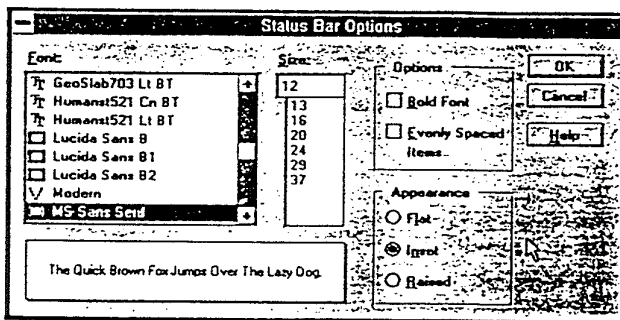


Figure 26.44
The Status Bar
Options dialog box.

Like the Button Bar Options dialog box, you can pick the font and font size in which you want the text on the Status Bar displayed from the Status Bar Options dialog box. The default font is the most readable, but WordPerfect offers you yet another personal preference here, and you might want to experiment with this option.

- The Status Bar Options dialog box also determines how the buttons on the Status Bar look. In the Appearance section, you can choose between Flat, Inset, or Raised buttons. The option you choose for your own workspace is a matter of aesthetics. Leave the appearance set to the default—Inset—for now. Click on OK to return to the Status Preferences dialog box.
- Click on the Time box, and then drag it into place after the Position box. Release the mouse button. Drag the Date box into place after the Time box and release the mouse button.
- Click on OK to return to the document screen. Double-click on the Date box on the Status Bar. The date appears in the document without typing (see fig. 26.45).

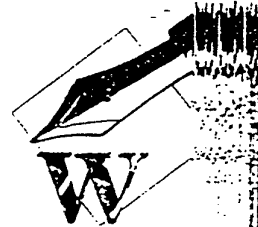
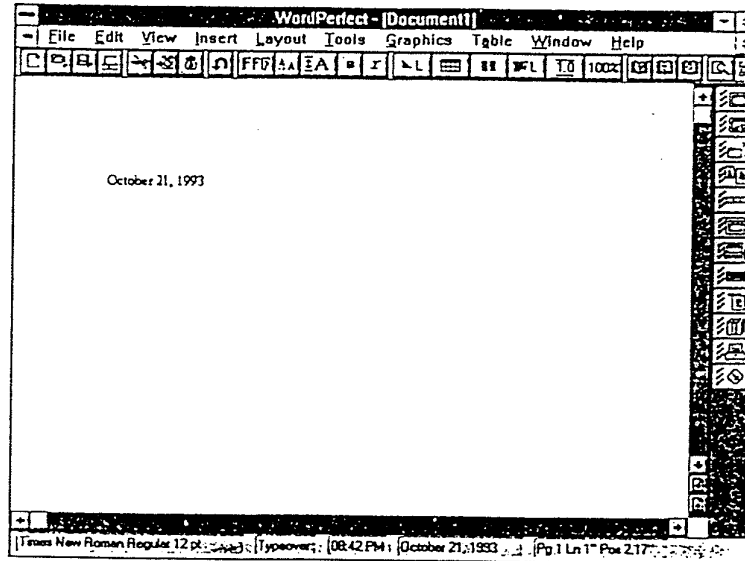


Figure 26.45

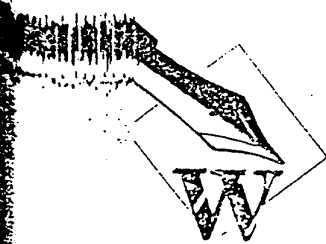
Double-click on the Date button on the Status Bar to insert the date into your document.



To return to the default setting for the Status Bar, choose Default from the Status Bar Preferences dialog box. If you later want to return to a previously constructed custom arrangement, you must reconstruct it because you cannot save multiple Status Bar configurations.

Hiding the Bars

You can see the considerable power and convenience built into WordPerfect's bars. The bars all can appear on-screen at the same time to serve you, but used together they take up much screen space and can hinder your view of a document. The good news is that with WordPerfect 6.0 for Windows, you can choose to display all, some, or none of the bars.



IBM

OS/2

Developer

\$9.95 U.S.
Display Until
7/15/93

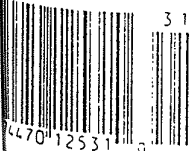
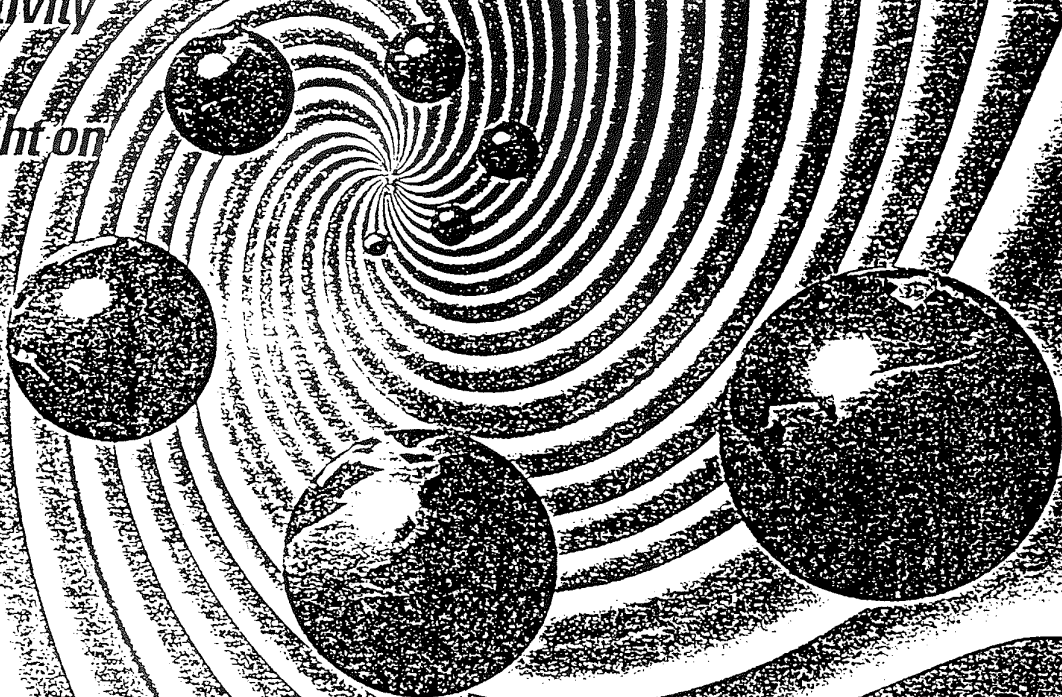
THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

Vol. 5 No. 2

**Communications
And
Connectivity**

**Spotlight on
Oracle
Corp.**

**More
On
SOM**



GUI

This article builds on "Demystifying Custom Controls" (Winter '93), explaining in more detail issues confronting the design and implementation of a custom control. BY MARK A. BENGE and MATT SMITH

Designing Custom Controls

IN THE LAST ISSUE, WE PRESENTED A graphical button that contained image and text components. We quickly explained some of the design and implementation issues that concerned the control. In this installment of the article, we will explain in more detail the issues confronting the design and implementation of a custom control.

FIRST PRINCIPLES

Most new controls are a clean slate, or "tabula rasa," when they are first conceived. A control is just another window. It can be thought of as a device for displaying information for, and gathering input from, users. Custom controls are not that hard to create as long as you have the proper information. Figure 1 shows four simple but useful controls not presently found in the OS/2 Presentation Manager.

The 3-D separator, the simplest control in Figure 1, uses the `GpiLine` function to create shadow and highlight lines. Figure 2 shows the source code that implements painting the control. (This and other sample source code fragments shown in this article can be found on CompuServe in LIB13 of the OS2DF2 forum, or through other electronic means outlined at the end of the article.)

The first component of the separator, the upper line, is drawn in the shadow color, while the second component, the lower line, is drawn in white, causing the three-dimen-

sional effect. There's not much to it, especially since the control does not interact with the user through either the keyboard or the mouse. The sample code provides for two styles, vertical and horizontal. Through other processing in the control, the rectangle coordinates of the control are determined and that value is used to determine the starting and ending points of the lines.

The pattern control (the halftone area in Figure 1) uses the `GpiPattern` function along with the draw and fill capabilities of the `GpiBox` function, as shown in Figure 3. Again, the painting of the control is not complex. The sample control allows for each of the different `Gpi` pattern types as a style. The painting routine determines the pattern style by querying the control style with `WinQueryWindowULong` with an index of `QWL_STYLE` and masking out the styles that are not valid `PATSYM_*` values.

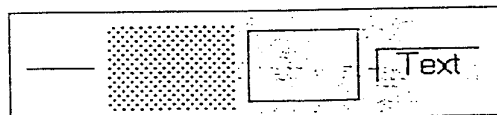


Figure 1. Example of custom control

The 3-D frame control, the third control from the left in Figure 1, uses the principles demonstrated in the 3-D line control. `GpiBox` is used to draw the two boxes, first the light box and then the shadow box. The boxes are offset from each other by one pixel, giving a



Mark Bengé



Matt Smith

```

        /* Paint the Control          */
case WM_PAINT :

        /* Get the address of the text from */
        /* the control's reserved memory */

    plf = (PLINEFIELD)WinQueryWindowULong(hWnd,
                                           QWL_USER);

    hPS = WinBeginPaint(hWnd, (HPS)NULL,
                       (PRECTL)NULL)
    GpiSetColor(hPS, SYSCLR_SHADOW);

    GpiMove(hPS, plf->aptl);
    GpiLine(hPS, &plf->aptl[1]);

    GpiSetColor(hPS, CLR_WHITE);

    GpiMove(hPS, &plf->aptl[2]);
    GpiLine(hPS, &plf->aptl[3]);

    WinEndPaint(hPS);
    break;

```

Figure 2. 3-D separator control painting code fragment

```

        /* Paint the Control          */
case WM_PAINT :

        /* Get the address of the control info from */
        /* the control's reserved memory */
        ppat = (PPATTERN)WinQueryWindowULong(hWnd,
                                              QWL_USER);

        /* Get the presentation space for the window */
        /* and draw the grid which the buttons are */
        /* placed */
    hPS = WinBeginPaint(hWnd, (HPS)NULL, (PRECTL)NULL)
    GpiSetPattern(hPS,
                 (LONG)WinQueryWindowULong(hWnd,
                                             QWL_STYLE) &
                 0x5f);

    ptl.x = ptl.y = 0L;
    GpiMove(hPS, &ptl);
    ptl.x = ppat->rcl.xRight;
    ptl.y = ppat->rcl.yTop;
    GpiBox(hPS, DR0_FILL, &ptl, 0L, 0L);
        /* Release the presentation space */
    WinEndPaint(hPS);
    break;

```

Figure 3. Pattern control painting code fragment

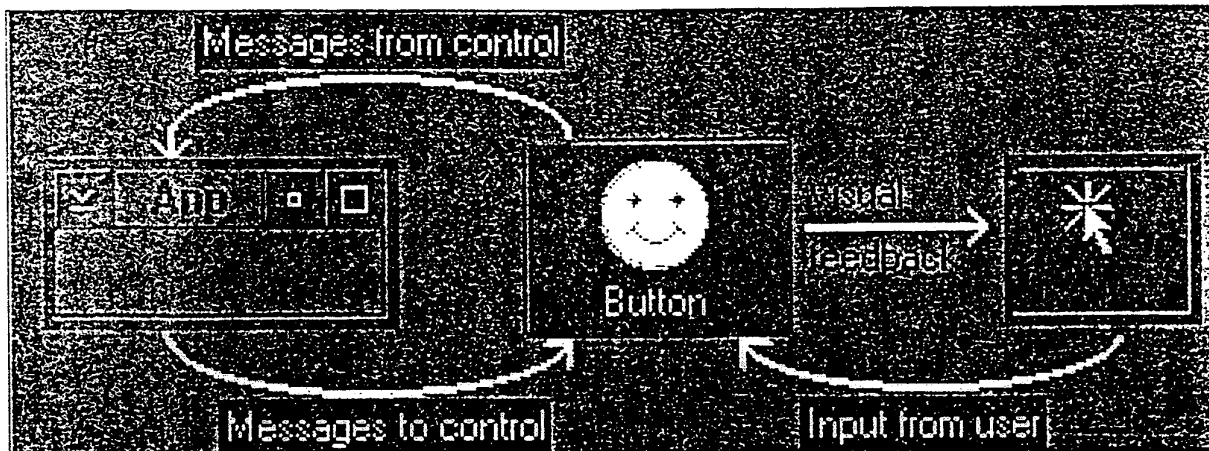


Figure 4. Control interaction diagram

three-dimensional effect.

Finally, the 3-D text, the last control in Figure 1, is a composite control consisting of a static text window and an outer line. Any special presentation parameter processing is done when the text window is created; this ensures that the text is

Understanding the role of a control allows you to determine how it should operate with users and the owning application.

in the color and font required. The 3-D frame is then drawn around the text window using a technique similar to that described for the 3-D frame. This time, the left and top edge are drawn in the shadow color with `GpiPolyLine`; the color is then changed to white before `GpiPolyLine` is used to draw the right and bottom edges. This causes the text to appear as though within a sunken area.

Special consideration is given to

the handling of messages; most messages of interest, such as `WM_PRESPARAMCHANGED`, need to be passed to the text window with `WinSendMsg`.

The architecture of the source code of the example controls is similar to that of the others. Its architecture is the foundation of a base control (which, in Presentation Manager terms, begins with the basic static control type).

CONTROL TYPOLOGY

The first of the basic control types is the static control, used basically to display information. Static controls found in Presentation Manager include text, group boxes, frames, icons, and bitmaps. These only display information; they allow for user input through neither the keyboard nor the mouse pointer.

Static controls are the easiest to implement; because they are concerned only with displaying information, they don't need a well-defined communication mechanism between the control and its owner.

The second type of control, however, must consider input. Through the mouse or keyboard, users can manipulate a control to inform the owning application of a specific selection. Examples of this type of

control are the button, slider, scroll bar and value set controls of Presentation Manager, which display only visual information such as scale, position, color, and so on. Users make a selection decision based upon the visual information presented by the controls.

A vertical scroll bar, for example, consists of a scroll button, shaft, and elevator or thumb. When the thumb is just below the top scroll button, the top scroll button is disabled, indicating to the user that the only direction of movement is down. Users can either click the mouse pointer on the down scroll button, click the mouse on the shaft below the thumb, or select the thumb and drag it downward toward the bottom scroll button. This type of operation helps users understand certain conditions within the application and allows the user to inform the application of a desire to change from a current condition to a new one, a structure that can be thought of in terms of action and reaction.

The third type of control, a combination of the first two types, must extend input to allow for selections; information displayed can be visually scanned and selected. The list box, combination box, and spin button are good examples of this type of control.

Message	Buttons	Container	Combo Box	Entry Field	List Box	Note Book	Slider	Spin Button	Static	Value Set
WM_ADJUSTWINDOWPOS			•	•	•					
WM_BUTTON1DBLCLICK	•	•	•	•	•	•	•	•		•
WM_BUTTON1DOWN	•	•	•	•	•	•	•	•		•
WM_BUTTON1UP	•	•	•	•	•	•	•	•		•
WM_BUTTON2DBLCLICK				•			•			
WM_BUTTON2DOWN		•		•	•		•			•
WM_BUTTON2UP		•					•			•
WM_BUTTON3DBLCLICK				•						
WM_BUTTON3DOWN		•		•	•					
WM_CHAR	•	•	•	•	•	•	•	•		•
WM_CLOSE						•				
WM_CONTROL	•	•	•	•	•	•	•	•		•
WM_CONTROLPOINTER		•	•			•	•			•
WM_CREATE	•	•	•	•	•	•	•	•	•	•
WM_DESTROY	•	•	•	•	•	•	•	•	•	•
WM_DRAWITEM		•				•	•		•	•
WM_ENABLE	•		•	•	•		•	•		•
WM_ERASEBACKGROUND		•								
WM_HELP		•				•				•
WM_HITTEST			•						•	
WM_HSCROLL		•			•					
WM_MATCHMNEMONIC	•									
WM_MOUSEFIRST				•	•					
WM_MOUSELAST				•	•					
WM_MOUSEMOVE	•	•	•	•	•	•	•	•	•	•
WM_MOVE			•							
WM_PAINT	•	•	•	•	•	•	•	•	•	•
WM_PRESPARAMCHANGED	•	•	•	•	•	•	•	•	•	•
WM_QUERYCONVERTPOS	•	•		•	•			•	•	
WM_QUERYDLGCODE	•			•					•	
WM_QUERYWINDOWPARAMS	•		•	•			•		•	•
WM_SETFOCUS	•	•		•	•	•	•	•	•	•
WM_SETSELECTION			•	•				•		
WM_SETWINDOWPARAMS	•		•	•			•		•	•
WM_SIZE		•					•	•		
WM_SYSCOLORCHANGED	•			•	•				•	•
WM_SYSVALUECHANGED	•			•						
WM_TIMER		•	•	•	•		•	•		
WM_TRANSLATEACCEL						•				
WM_VSCROLL		•			•					
WM_WINDOWPOSCHANGED	•		•	•	•				•	

Table 1. Messages handled by Presentation Manager control

Message	Use
BM_CLICK	Allows the owning application to simulate the clicking of pushbutton
BM_QUERYCHECK	Used to determine button check state
BM_QUERYCHECKINDEX	Used to determine the index of the button checked within a group of buttons
BM_QUERYHILITE	Used to determine if a button is highlighted
BM_SETCHECK	Sets the check state of a button
BM_SETDEFAULT	Sets the button state to its default condition
BM_SETHILITE	Sets the highlight state of the button

Table 2. Button-specific messages

```
typedef struct _BTNCDATA /* btncd */
{
    USHORT cb; /* Structure Size */
    USHORT fsCheckState; /* Button Check State */
    USHORT fsHiliteState; /* Button HighLite State */
    LHANDLE hImage; /* Icon/Bitmap Handle */
} BTNCDATA;
```

Figure 5. Button CTLDATA structure

Message	Use
BN_CLICKED	Button clicked on
BN_DBLCLICKED	Button has been doubleclicked on
BN_PAINT	User drawn button requires painting

Table 3. Button notification messages

The listbox control within Presentation Manager is comprised of a list area and scroll bar. The scroll bar operates like the one described previously. The list area operates as a group of static text controls. When combined, the actions of users on the scroll bar determine the contents of the list area. As users cause the scroll bar to change position, through one of several

means (scroll button, page up and down, dragging), the list area is scrolled in conjunction with it. A further interaction allows users to select an item within the list area, which usually indicates to the application that a user wants to manipulate or use that item.

The final type of control uses all the techniques of the others, plus allowing editing of displayed infor-

mation. Examples of this type are the Presentation Manager entry field and the multiple-line entry field. Here, the control displays information and allows it to be selected and edited.

A control must be able to:

- Display some type of information
- Accept input if it is to denote a selection or manipulation
- Allow the entering or editing of information.

Understanding the role of a control allows you to determine how it should operate with users and the owning application. The control interacts with users both visually and through the keyboard and mouse, allowing interaction with the application through various messages. The application may also communicate with the control through messages. Figure 4 shows this interaction process.

In all cases, the control displays some information. As shown in Figure 4, the control acts in a sort of isolation: it is an entity that interacts through a defined set of conventions with the application and the user. The control must be designed to assume that it cannot depend on the application for necessities such as data in its life-span.

The control must be created to work from the desktop as though no application owned it. Although this idea may seem strange, it allows controls to work under almost any condition. It is possible for the control to be created with the desktop as the parent and owner. This means that the communication flow for control-specific messages is only from the control to the desktop. The messages most likely won't be understood by the desktop, and the desktop will not send any control specific messages to the control, because it has no

```

WinCreateWindow(HWND hwndParent, PSZ pszClass,
                PSZ pszName, ULONG flStyle, LONG x,
                LONG y, LONG cx, LONG cy,
                HWND hwndOwner, HWND hwndInsertBehind,
                ULONG id, PVOID pCtldata,
                PVOID pPresParams);

```

Figure 6. WinCreateWindow definition

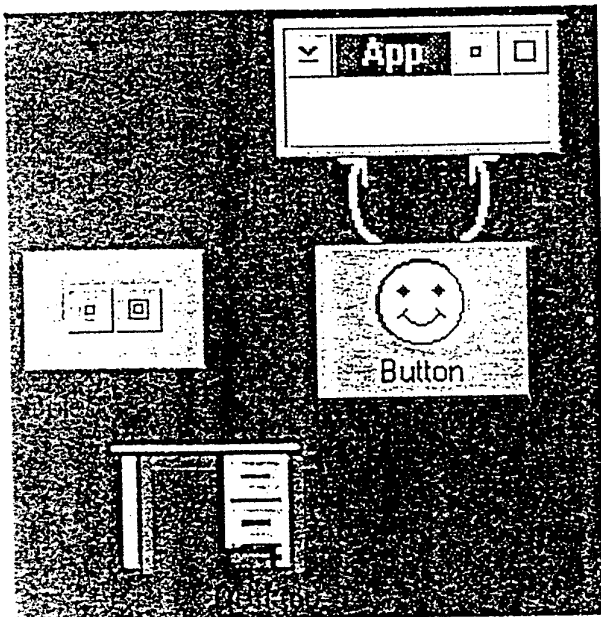


Figure 7. Application parent-owner-control relationships

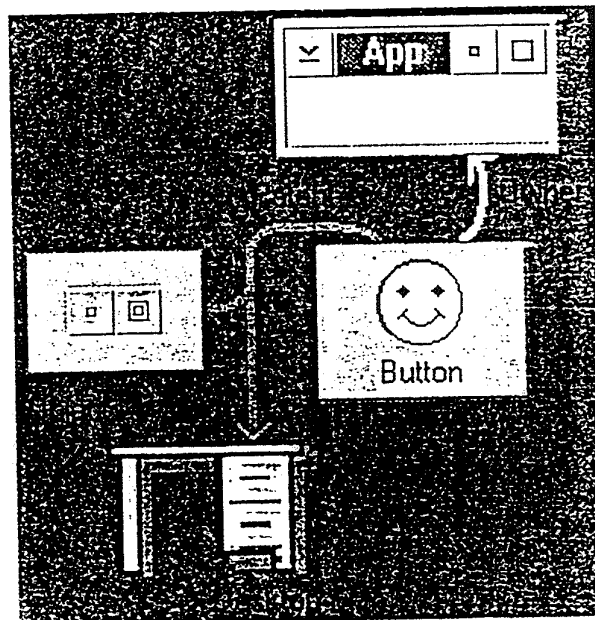


Figure 9. Desktop/object window parent-owner-control relationships

understanding of the control's message conventions.

Controls should be designed to get all required resources during creation and perform as much setup as possible without further messages from their creator. Generally, this is done through the CTLDATA mechanism, part of the WinCreateWindow function shown in Figure 6 (parameter 12, PVOID pCtldata). While this use of CTLDATA may be sufficient during creation, care should be taken so if the data structure is not present, messages to the control with the appropriate resources can achieve the same effect, still allowing the control to operate even in a minimal state.

While designing the control, continually remind yourself how to maintain its responsiveness. You

can do this through prudent coding of sections within the control, perhaps with the use of threads. Delays in responsiveness will cause the control to respond unevenly, somewhat like the notebook control in OS/2 2.0.

WHAT'S IN A DESIGN?

A control can be as simple as a line or pattern and as complex as the container control. It need not be a unique entity, but can be a composite control, made up of other controls, similar to the 3-D text control presented earlier.

You have to plan and architect a control's appearance and how it is to interact with the application and with users. This includes the external data (CTLDATA or messages) passed to the control through stan-

dard functions like WinSetWindowText or WinSetDlgItemText, as well as special-purpose messages intended for use only by the control.

MESSAGES AND MORE MESSAGES

One of the major design issues to contend with is the handling of messages sent by Presentation Manager to your control. Unfortunately, very little in the OS/2 Technical Library mentions the creation and handling of custom controls. Table 1 outlines the messages handled by the various Presentation Manager controls; you can guide the message handling design of your control by comparing your control's characteristics with those of the Presentation Manager controls.

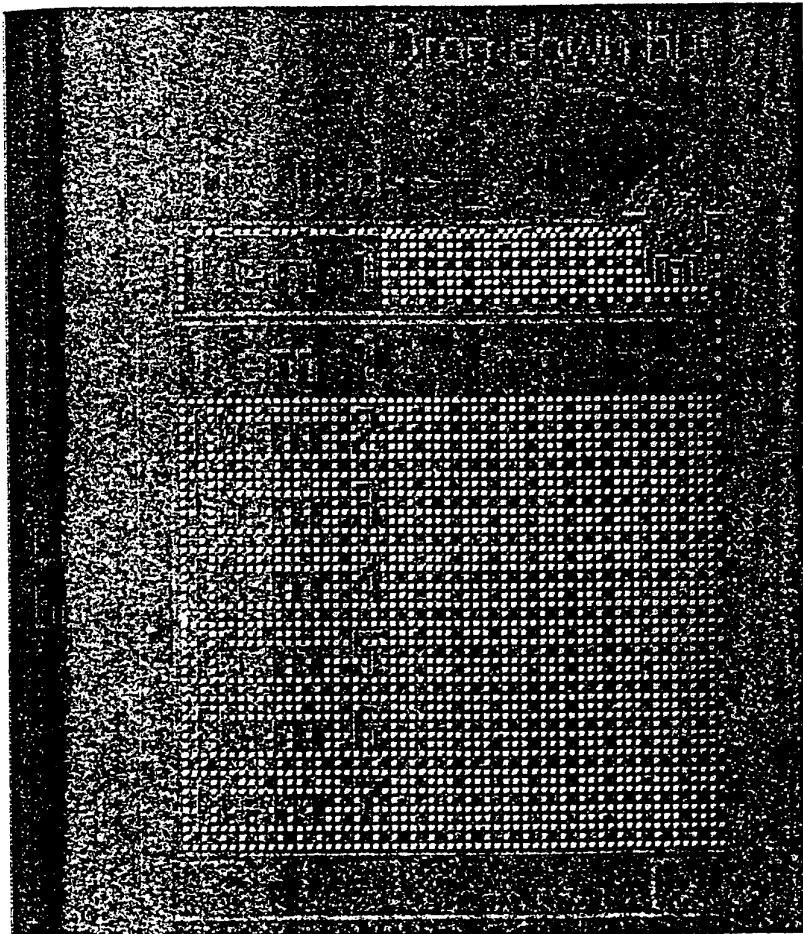


Figure 8. Combo box

```

typedef struct _CREATESTRUCT /* crst */
{
    PVOID pPresParams; /* Pres. Parameters Pointer */
    PVOID pCtlData; /* Control Data Pointer */
    ULONG id; /* ID Value */
    HWND hwndInsertBehind; /* Z-Order Placement */
    HWND hwndOwner; /* Owner Window Handle */
    LONG cy; /* Window Height */
    LONG cx; /* Window Width */
    LONG y; /* Window Vert. Placement */
    LONG x; /* Window Horz. Placement */
    ULONG flStyle; /* Window Style */
    PSZ pszText; /* Window Text */
    PSZ pszClass; /* Class */
    HWND hwndParent; /* Parent Window Handle */
} CREATESTRUCT;

```

Figure 10. CREATESTRUCT structure

1 It is recommended that you be prepared to handle all messages received by your control. In reality, however, you should handle only those messages that will affect your control. For example, you should handle the WM_PAINT message, whereas you may need to handle the WM_SETSELECTION message. Finally, if you want to create a composite control, you may have to handle those messages in the subclass procedure of a control that you are using to create your control.

15 After deciding which messages you wish to handle that will be specifically sent to the control by Presentation Manager, you need to define messages sent by the owning application. These messages are control-defined and unique to the control. Table 2 shows the list of such messages defined for Presentation Manager buttons.

25 **Message Use.** These messages should be coordinated with the CTLDATA structure where possible. Figure 5 shows the corresponding Presentation Manager button CTLDATA type structure, BTNCDATA.

The message BM_SETCHECK corresponds to the fsCheckState field, while the BM_SETHILITE message responds to the fsHiliteState field.

35 In conjunction with the WM_CONTROL message are the notification messages that the control uses to notify the owning application of a significant event. Table 3 shows the list of notification messages used by Presentation Manager buttons.

THE FAMILY TREE

45 To understand the relationship between the family tree and the control, you need to understand how the control is ultimately created. All windows are eventually created by WinCreateWindow, defined in Figure 6. The first parameter is the parent window handle and the ninth parameter is the owner window handle. Usually, the parent and the owner are the same for

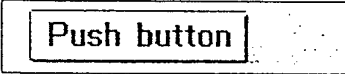
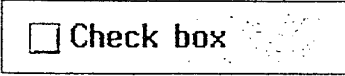
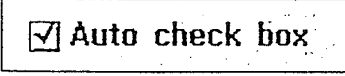
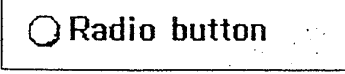
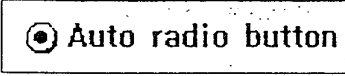
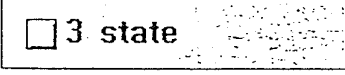
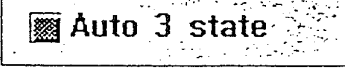
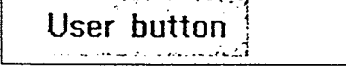
Base Style	Button
BS_PUSHBUTTON	
BS_CHECKBOX	
BS_AUTOCHECKBOX	
BS_RADIOBUTTON	
BS_AUTORADIOBUTTON	
BS_3STATE	
BS_AUTO3STATE	
BS_USERBUTTON	

Table 4. Button base styles

```
typedef struct _USERBUTTON /* btn */
{
    HWND    hwnd;          /* Button Window Handle */
    HPS     hps;           /* Presentation Space Handle */
    ULONG   fsState;       /* Current State */
    ULONG   fsStateOld;    /* Previous State */
} USERBUTTON;
```

Figure 11. USERBUTTON structure

controls, as shown in Figure 7. However, depending on how the control works, the owner and the parent could be different, as shown in Figure 9.

You can specify the owner and parent to be the same through the `WinCreateWindow` function; in this case, messages are sent to the owner window and the control is

displayed when the parent is displayed.

You can also use `WinCreateWindow` to specify differing parents and owners for the control. A good example of this is the combo box, shown in Figure 8, in which a composite control is comprised of three components: an entry field, a list box, and a drop-down button.

When the control is first created (assuming the style is drop-down or drop-down list), the entry field's owner and parent are the drop-down button. The list box used to display the drop-down list is created with the owner the drop-down button and the parent `HWND_OBJECT`. This relationship is shown in Figure 9.

This initially causes the list box component to be hidden, because descendants of object windows are not visible. Messages (`LM_*`) sent to the combo box handle are redirected to the listbox.

When a user clicks the mouse pointer on the drop-down button, the combo box changes the parent of the list box from the object window to the desktop, which allows it to be displayed. When a user selects an item within the list box, the notification message is sent to the drop-down button that owns it, not to the desktop (which is its parent). The message is then repackaged and sent on to the application that owns the combo box.

When you send a message back to the owning window, you send it to the owner handle. The owning application can set the parent to an object window while you are still processing information; in this case, the owner would still be an application window that receives all messages from the control. When the drop down button is pressed again, the list box can be hidden by changing the parent from the desktop back to the object window.

LIFE OF A CONTROL: AN EXAMPLE

The best way to understand control design is to dissect an existing control. We will now look at the Presentation Manager button control and explain what is happening under the covers. This will help you both understand the processing of existing Presentation Manager controls and model your control on them.

Additional Style Use

- **BS_AUTOSIZE** Button is sized to contents to ensure that contents are visible.
- **BS_BITMAP** Button will display graphic bitmap image supplied through text (#id) or through hImage of BTNCDATA structure. (This support was not implemented in OS/2 2.0 GA, but was implemented in the SP and OS/2 2.1.)
- **BS_DEFAULT** Push button is drawn with heavier border indicating button will automatically be selected if the ENTER key is pressed.
- **BS_HELP** Push button will post a WM_HELP message instead of the normal WM_COMMAND.
- **BS_ICON** Button will display graphic icon image supplied through text (#id) or through hImage of BTNCOTA structure.
- **BS_NOBORDER** Push button will be displayed with no border.
- **BS_NOCURSORSELECT** Auto radio button will not select itself when given the focus through the keyboard.
- **BS_NOPOINTERFOCUS** Focus will not be received by the button though the button is still selectable. This is generally used with the help pushbutton so that the currently focused control will be used to look up help from the window or dialogue subtable .
- **BS_SYSCOMMAND** Push button posts a WM_SYSCOMMAND instead of the normal WM_COMMAND message.

Table 5. Additional style use

WM_CREATE. Every control has three stages of life: a beginning, a middle, and an end. WM_CREATE, the beginning of a control's life, is received when the control is created through WinCreateWindow.

Two structures are presented through the mp1 and mp2 message parameters. The first structure, passed in the mp1 message parameter, corresponds to the PVOID pCtlData parameter of WinCreateWindow. The second structure, passed in the mp2 message parameter, contains a pointer to the creation structure shown in Figure 10. This structure helps you determine the parent and owner of the control, along with its internal ID value, style, text, position, and size. You will want to save most of this informa-

tion internally within a private data structure. You should save the data structure pointer within the window words of the control, which you have reserved with the WinRegisterClass function. The data is assembled from the parameters of WinCreateWindow. The elements of the CREATESTRUCT structure map to the parameters from right to left, as in the prototype in Figure 6.

The Presentation Manager button control determines if the BTNCDATA structure, shown in Figure 5, has been passed to it by checking if mp1 is NULL or contains a valid pointer. It then verifies that the structure is valid by checking its size, which is the first word of the structure. Thus, the first word or long value of the CTLDATA type

structure must contain its size. This value is used by OS/2 Presentation Manager to allocate memory for the structure before it is passed to the control.

Any relevant values within the BTNCDATA structure are recorded internally within the control's private data area to allow the correct representation of the control. General setup, such as transferring a copy of the text of the button to internal memory, is then performed.

If anything is incorrect, the control returns TRUE to indicate to the window manager that the creation of the control should stop. This may cause a domino effect depending on who is creating the control. If the control is being created by the

Dialogue Code	Meaning
DLGC_RADIOBUTTON	Identifies button as a radio button
DLGC_DEFAULT	Identifies button as default push button
DLGC_PUSHBUTTON	Identifies button as a non- default push button
DLGC_CHECKBOX	Identifies button as a check box

Table 6. Dialogue codes

```

typedef struct _WNDPARAMS      /* wprm          */
{
    ULONG fsStatus;           /* Query/Set Flag      */
    ULONG cchText;           /* Text Size           */
    PSZ pszText;            /* Text Pointer        */
    ULONG cbPresParams;      /* Pres. Parameters Size */
    PVOID pPresParams;      /* Pres. Parameters Pointer */
    ULONG cbCtlData;        /* Control Data Size   */
    PVOID pCtlData;        /* Control Data Pointer */
} WNDPARAMS;

```

Figure 12. WNDPARAMS structure

dialogue manager through a call to `WinCreateDlg`, `WinDlgBox`, or `WinLoadDlg`, the entire dialogue creation will fail and the dialogue will not be displayed. Unfortunately, this behavior is not very well documented.

and the memory for private data is released back to the system. Also, any icons or bitmaps loaded by the control as part of the control's text string would be destroyed here.

WM_PAINT. When the button needs to show itself to the world, it occurs during the processing of the `WM_PAINT` message. The control determines the current style by checking through `WinQueryWindowU-Long` with the `QWL_STYLE` index to ensure that the control is painted correctly. The styles shown in Tables 4 and 5 are used as the basis for most painting. Using this information, the button then checks to see if it is visible; if it isn't, the rest of the processing is ignored, since it is unnecessary.

The button is painted using the bitmap, line, and text graphics calls. The bitmap calls are used to draw the radio button and check box components—unless it is a push

button, in which case the outline edges are drawn with the line functions. The text for the button is also drawn, including the mnemonic if one has been provided.

For the user button style, the button fills in a `USERBUTTON` structure, shown in Figure 11, that allows the owner of the control to paint the button in a manner specific to the owning application. A `WM_CONTROL` message is packaged with the `BN_PAINT` notification and is sent to the owner of the button.

WM_WINDOWPOSCHANGED. When the button changes position or size, this message is received and used to force a recalculation of the button dimensions.

WM_SETFOCUS. When the button receives the focus, the mouse pointer is setup for use by the button. This setup involves the retrieval of the default arrow shape. When the

Every control has three stages of life, a beginning, a middle, and an end.

With the normal return value of `FALSE`, the creation process will continue.

WM_DESTROY. The button control ends its existence when it receives the `WM_DESTROY` message. Usually, any clean up is performed here,

focus is being lost, it discards the mouse pointer.

WM_MOUSEMOVE. When using the mouse pointer setup through the `WM_SETFOCUS` message, the mouse pointer is displayed over the control after the button has sent a `WM_CONTROLPOINTER` message to the owning application, giving it a chance to change the pointer shape.

When the mouse is in capture mode after a `button 1 down` event, the mouse position is monitored to see if it is within or outside the button limits. When it is within the limits, the button is displayed in the highlighted state, and when it is outside the limits, the button highlight state is removed.

WM_BUTTON1UP. The button control ignores this message if the mouse capture is not active. When capture is active, it is removed and a `BN_CLICKED` notification message is packaged for the `WM_CONTROL` message that is sent back to the owner of the button. However, if it was a double click, `BN_DBLCLICKED` is packaged for the `WM_CONTROL` message.

WM_BUTTON1DBLCLICK. The event is recorded to allow the `BN_DBLCLICKED` notification to be sent after the `WM_BUTTON1UP` event to the owner of the button.

WM_BUTTON1DOWN. When `button 1` of the mouse is pressed, the mouse is placed in capture mode to allow the monitoring of mouse movement. When `button 1` is released, it is recorded by the button control. The capture is necessary so that if the user moves the mouse pointer outside the limits of the control, all mouse-related events are still received by the button control and the proper notification is performed.

WM_CHAR. Keyboard presses are handled by the button only when

no mouse capture is being performed. The idea here is that actions should not be confused with the function of the control. This situation is similar to serialization of a resource through semaphores. Selection keys such as the spacebar, as well as the Tab and cursor movement keys, are processed.

WM_QUERYDLGCODE. When Presentation Manager is initializing the control, it sends this message to interrogate the control about its capabilities. The button control returns `DLGC_BUTTON` or'ed with one of the values shown in Table 6. These codes allow the Presentation Manager to determine how to handle certain events, including mnemonic selection and decoding.

WM_QUERYWINDOWPARAMS. This message is received by the button control when the `WinQueryWindowText` or `WinQueryDlgItemText` functions are used. The `WPM_TEXT` code is used within the `WNDPARAMS` structure, shown in Figure 12, through the `fsStatus` field. It is also received by the button control when the `WinQueryWindowTextLength` or `WinQueryDlgItemTextLength` functions are used. The `WPM_CCHTEXT` code is used within the `fsStatus` field.

WM_SETWINDOWPARAMS. This message is received by the button control when the `WinSetWindowText` or `WinSetDlgItemText` functions are used. The `WPM_TEXT` code is used within the `WNDPARAMS` structure, shown in Figure 12, through the `fsStatus` field.

WM_PRESPARAMCHANGED. When any of the presentation parameters are changed either by Presentation Manager through a scheme palette change or by the owning application through a `WinSetPresParams` or `WinRemovePresParams` function, the button control forces a repaint of the entire control so the control

reflects the changes made.

WM_ENABLE. The enabling and disabling of the control through the `WinEnableWindow` function causes the button to set an internal flag to allow the control to determine when to ignore keystrokes. It also causes the button to repaint itself to reflect the state change.

WM_SYSCOLORCHANGED. When a user changes the scheme palette for the entire system, this message is received by the button control. The button then causes a repaint to

Use CTLDATA structures that will allow the formal setup of data through the creation of the control as much as possible.

occur, reflecting any changes made to the system colors that may have been used by the button.

WM_MATCHMNEMONIC. If a mnemonic has been provided in the text of the control (designated by a `^` before the character that is to act as the mnemonic), the value passed through the `mp1` message parameter is checked as the designated character. When a match occurs, a value of `TRUE` is returned. However, when no match occurs a value of `FALSE` is returned. Presentation Manager then uses the return value to determine if it should send the `BM_CLICK` message to the button control to simulate a button click.

WM_QUERYCONVERTPOS. The button returns a `QCP_NOCONVERT` to indicate that no conversion should

occur with double-byte character set (DBCS) processing.

GENERAL RULES

The following section summarizes most of the rules you need to be concerned with when creating custom controls. Further information on each topic can be found, where indicated with "(Winter '93)," in the Winter 1993 *OS/2 Developer* article "Demystifying Custom Controls," by Mark Bengé and Matt Smith, and where indicated with "(Spring '93)," in this article.

Use `CTLDATA` structures that will allow the formal setup of data through the creation of the control as much as possible. The information contained within the structure

The complexity of a control is constantly misunderstood... controls can be very simple in purpose and implementation.

should correspond to messages that can be sent to the control. This may be important in allowing compatibilities to third-party products and other tools. (Winter and Spring '93)

The `CTLDATA` structure's first word *must* contain the size of the entire structure. This is a cardinal rule that must not be broken within *OS/2 2.0*. (Winter and Spring '93)

Use dialogue units as the basis of measurement for the control externally and pixels internally. This is important if you want the control to work and look similar over different screen resolutions.

Something that is three pixels on a VGA system may appear as a dot on a 1280x1024 system.

Maintain a clean and abstract interface so controls that allow the setting of units are based on a range of values. This is similar to the way scroll bars allow you to set the starting and ending points of the range (unlike sliders, which are based on an absolute zero-based system). While this may make the internals of the control more complex, the external use of the control will be easier because the control calculates and reports the position—not the window or dialogue that owns the control.

Maintain a cache of information that allows for speedy painting and keyboard and mouse responsiveness. (Winter and Spring '93)

Keep the control simple and straightforward. Try not to overload your initial controls with functionality by trying to make a control that exceeds all others. Iteratively build up your controls in complexity as you gain experience; it won't take long and they will all work and be useful. (Spring '93)

Keep the control as tight and compact as possible to ensure that it is responsive to user actions. (Spring '93)

When painting, set presentation parameters in RGB mode instead of color index mode. Although it's more difficult, using RGB mode gives you more color flexibility. (Winter '93)

If the control is providing for multiple items, as with a list box, allow for a data pointer that the control owner can reference through a message (for example, list box messages `LM_SETITEMHANDLE` and `LM_QUERYITEMHANDLE`). This makes it easy for users to reference data associated with a selected item.

Presentation parameters are handled on your behalf by Presentation Manager; you need to monitor changing of the parameters only

for use in painting routines.

Fonts and foreground and background colors are handled on your behalf by the owning application window or dialogue. Only areas outside the defaults, such as line type and pattern type, need to be handled. (Winter '93)

When registering a class, reserve room for a private data pointer along with a four-byte pointer area that can be used by the owner of the control. This area should be referenced with the `QWL_USER` manifest constant. The private data for the control should be referenced with `QWL_USER + 4`. The total reserved memory size is eight bytes. (Winter '93)

Ensure that the control has a default style and a default set of limits where appropriate. Assume that the control could be created without any specified styles and limits. An example of this is the Presentation Manager entry field, with its default style of `ES_LEFT` and default text limit of 32 characters.

CONCLUSION

The complexity of a control is constantly misunderstood. As demonstrated with the sample controls in Figure 1, controls can be very simple in purpose and implementation. Don't be intimidated from building a control simply because you think it is complex. A control is just another window, and an application can be thought of as one big composite control.

In a future article, we will develop a control that from which you can select RGB colors, based on the principles of the color wheel. This control will introduce `Gpi` functions and techniques for clipping, bitmap drawing, and threads to maintain responsiveness. While we haven't yet documented how to use threads in this way, the information will be covered later.

Mark A. Bengé, IBM Programming Systems Laboratory, 11000 Regency Pkwy., Cary, N.C. 27512. Bengé is a senior associate programmer who joined IBM in 1989. He has worked on record-level I/O for OS/2 and CUA Controls Library/2. He now works in IBM class library user interface development, where he is implementing C++ classes for direct manipulation. Bengé received a B.S. in computer science from Western Carolina University.

Matt Smith, Prominare Inc., 100 Front Street East, Toronto, Ontario, Canada, M5A 1E1. Smith is lead architect for the Prominare Development System, an OS/2 2.0 advanced GUI development environment. He has been actively involved with OS/2 since 1988, co-founding Prominare in 1990. Smith has a degree in architecture from the University of Waterloo.

REFERENCES

OS/2 2.0 Programming Guide Volume II (IBM Doc. S10G-6494)

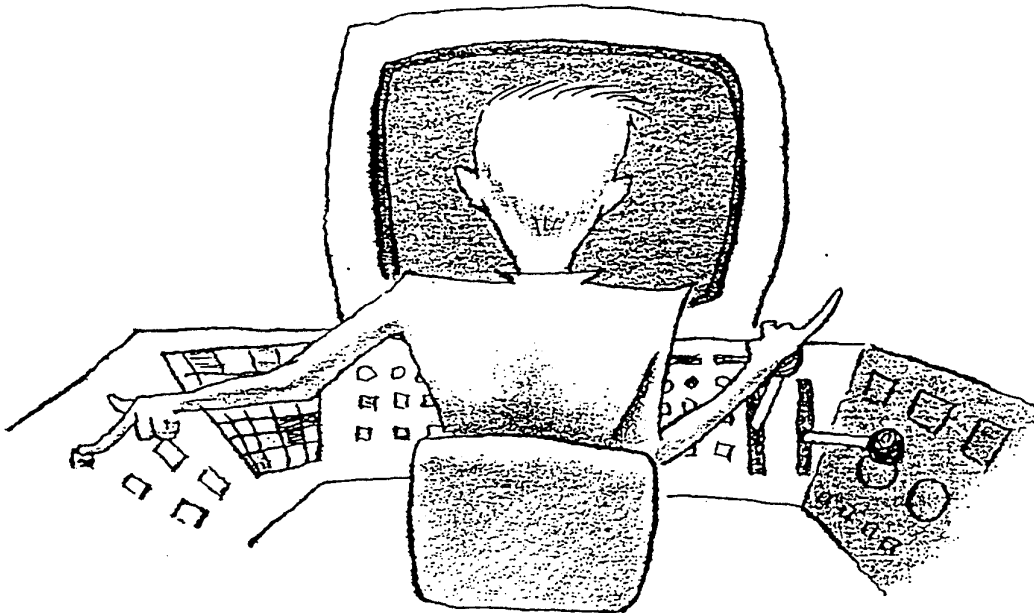
OS/2 2.0 Presentation Manager Programming Reference Volume I (IBM Doc. S10G-6264)

OS/2 2.0 Presentation Manager Programming Reference Volume II (IBM Doc. S10G-6265)

OS/2 2.0 Presentation Manager Programming Reference Volume III (IBM Doc. S10G-6272)

The source code can be downloaded from several electronic sources:

- In the United States, call the IBM PCC BBS at (404) 835-6600. The source code for this article is in File Area 11 under the name CTRLDES.ZIP.
- In Europe, you can find the source code on the International OS/2 Users Group BBS at +44 (0)454 633197 or +44 (0)454 633420.
- On CompuServe, the source code is in LIB13 of the OS2DF2 forum.
- If you have a VM account on an IBM node, you can receive the source code with the command `REQUEST CTRLDES FROM BANZAI AT CARVM3`.



INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 95/11025

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 584 392 (COHAUSZ) 2 March 1994 see the whole document ---	1-18
A	EP,A,0 483 777 (HEWLETT-PACKARD COMPANY) 6 May 1992 see column 13, line 55 - column 14, line 52; figures 6-9 see column 19, line 6 - column 20, line 13 ---	1-17
A	STEPHEN HARRIS ET AL.: 'Inside WordPerfect 6 for Windows' 1994, NEW RIDERS PUBLISHING, USA see page 1104, line 18 - page 1107, last paragraph; figures 26.42-26.45 --- -/--	1-17

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

* Special categories of cited documents:

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

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

Date of the actual completion of the international search

18 December 1995

Date of mailing of the international search report

03.01.96

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax (+31-70) 340-3016

Authorized officer

Fonderson, A

INTERNATIONAL SEARCH REPORT

national Application No
PCT/US 95/11025

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>IBM OS/2 DEVELOPER, vol.5, no.2, Spring 1993, US pages 72 - 85 MARK A. BENGE AND MATT SMITH: 'Designing Custom Controls' see page 74, middle column, line 15 - page 77, right column, line 27 see page 79, right column, line 49 - line 51; figure 6 see page 83, middle column, line 11 - line 21</p>	<p>1, 11, 15-18</p>

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 95/11025

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-0584392	02-03-94	NONE	
EP-A-0483777	06-05-92	AU-B- 640281	19-08-93
		AU-B- 8686491	07-05-92
		CA-A- 2054479	01-05-92
		CN-A- 1064557	16-09-92
		JP-A- 6012211	21-01-94



DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
D, Y	EP-A-0 352 741 (HEWLETT-PACKARD COMPANY) * column 3, line 2 - line 41 * * column 8, line 3 - line 12 * * column 10, line 23 - line 27; figure 4 * ---	1, 2, 10, 11, 14, 15	G06F3/033
Y	IEEE COMPUTER GRAPHICS AND APPLICATIONS, vol. 8, no. 5, September 1988, NEW YORK US pages 65 - 84; B. A. MYERS: 'A Taxonomy of Window Manager User Interfaces' * page 75, left column, line 33 - line 53; figures 1, 3 *	1, 2	TECHNICAL FIELDS SEARCHED (Int. Cl.5)
A	---	5	
A	RESEARCH DISCLOSURE, no. 307, November 1989, HAVANT GB page 901; 'Window Shades' Capability' * the whole document *	3, 4	G06F
A	RESEARCH DISCLOSURE, no. 305, September 1989, HAVANT GB page 651; 'Virtual Office Desktop Drawers' * the whole document *	6, 7	
Y	HEWLETT-PACKARD JOURNAL, vol. 40, no. 4, August 1989, PALO ALTO US pages 23 - 31; B. LAM ET AL.: 'The NewWave Office' * page 24, left column, line 30 - right column, line 3 * * page 29, right column, line 13 - line 34 *	1, 11, 14, 15	
A	---	8, 9	
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 19 JUNE 1992	Examiner Piero Bravo
<p>CATEGORY OF CITED DOCUMENTS</p> <p>X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document</p> <p>T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons</p> <p>* : member of the same patent family, corresponding document</p>			

EPO FORM 150 (03.92) (P0691)



DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
Y	IEEE, PROCEEDINGS OF WORKSHOP ON VISUAL LANGUAGES 27 June 1986, DALLAS, TEXAS, US pages 99 - 106; R.B. SMITH: 'The Alternate Reality Kit' * page 101, left column, line 33 - line 35 * * page 105, left column, line 13 - line 22; table 1 *	1, 10	
A	-----	13	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 19 JUNE 1992	Examiner Piero Bravo
<p>CATEGORY OF CITED DOCUMENTS</p> <p>X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document</p> <p>T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons</p> <p>* : member of the same patent family, corresponding document</p>			

EPO FORM 1501 (01/91) (P.04/91)



UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
A.W.M.S.

SERIAL NUMBER	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
---------------	-------------	----------------------	---------------------

08/316,237 09/30/94 CHRISTENSEN

S 04860 P1365
EXAMINER

DELA TORRE, C

ART UNIT PAPER NUMBER

24M1/0320
BLAKELY SOKOLOFF TAYLOR AND ZAFMAN
12400 WILSHIRE BOULEVARD
7TH FLOOR
LOS ANGELES CA 90025

5

2415
DATE MAILED:

03/20/96

This is a communication from the examiner in charge of your application.
COMMISSIONER OF PATENTS AND TRADEMARKS

This application has been examined Responsive to communication filed on _____ This action is made final.

A shortened statutory period for response to this action is set to expire 3 month(s), 0 days from the date of this letter.
Failure to respond within the period for response will cause the application to become abandoned. 35 U.S.C. 133

Part I THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:

- 1. Notice of References Cited by Examiner, PTO-892.
- 2. Notice of Draftsman's Patent Drawing Review, PTO-948.
- 3. Notice of Art Cited by Applicant, PTO-1449. 1 sheet
- 4. Notice of Informal Patent Application, PTO-152.
- 5. Information on How to Effect Drawing Changes, PTO-1474.
- 6. _____

Part II SUMMARY OF ACTION

1. Claims 1-18 are pending in the application.

Of the above, claims _____ are withdrawn from consideration.

2. Claims _____ have been cancelled.

3. Claims _____ are allowed.

4. Claims 1-18 are rejected.

5. Claims _____ are objected to.

6. Claims _____ are subject to restriction or election requirement.

7. This application has been filed with informal drawings under 37 C.F.R. 1.85 which are acceptable for examination purposes.

8. Formal drawings are required in response to this Office action.

9. The corrected or substitute drawings have been received on _____. Under 37 C.F.R. 1.84 these drawings are acceptable; not acceptable (see explanation or Notice of Draftsman's Patent Drawing Review, PTO-948).

10. The proposed additional or substitute sheet(s) of drawings, filed on _____, has (have) been approved by the examiner; disapproved by the examiner (see explanation).

11. The proposed drawing correction, filed _____, has been approved; disapproved (see explanation).

12. Acknowledgement is made of the claim for priority under 35 U.S.C. 119. The certified copy has been received not been received been filed in parent application, serial no. _____; filed on _____.

13. Since this application appears to be in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under Ex parte Quayle, 1935 C.D. 11; 453 O.G. 213.

14. Other

EXAMINER'S ACTION

Part III DETAILED ACTION

Specification

1. The disclosure is objected to because of the following informalities: on p. 21, line 23, after "has", --not-- should be inserted, and on p. 28, lines 11 - 12, the sentence is incomplete. Is it part of the previous sentence, or something else? Appropriate correction is required.

Claim Objections

2. Claim 5 is objected to because of the following informalities: "first window regions" should be singular. Appropriate correction is required.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. § 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1 - 18 are rejected under 35 U.S.C. § 102(b) as being anticipated by Mills et al. (U.S. patent 5,202,961).

Art Unit: 2415

As to independent claim 1, Mills et al., hereinafter Mills, rendered anticipated the following subject matter:

"a processor" with computer system 14, at Fig. 1, and col. 3, lines 22 - 26, 48 - 51;

"a data display screen" 18, at Fig. 1, and at col. 3, lines 23 - 24;

"cursor control device" 19, at Fig. 1, and at col. 3, line 24;

"a window generation logic" with the computer system 's software, at col. 3, line 66, to "generate and display a first window region" with view window 20, at Fig. 2, and at col. 3, line 68; and

"indicia generation logic" also with the computer system 's software, at col. 3, line 66, to "generate data for display in at least one display area in the first window" such as with video window 22, and control window 24, at col. 3, line 68 to col. 4, line 1, such that the "indicia generation logic use message-based communication to exchange information to coordinate activities" at col. 3, lines 22 - 26.

Mills also teaches that the first window region comprises a "control strip" [claim 2] with controller 36, in Fig. 2, and at col. 4, lines 32 - 40; wherein at least one display area is variably sized [claim 3] using size boxes 30, at Fig. 2, and at col. 4, lines 8 - 9; and that the size of the first window region is variable [claim 4] also at col. 4, lines 8 - 9.

In addition, Mills teaches sizing the first window region so that none of the display areas are visible [claim 5] with close box 28, at Fig. 2, and at col. 4, lines 7 - 8, or all [claim 6] or a portion [claim 7] of the display areas are visible, both at col. 4, lines 8 - 9.

Art Unit: 2415

Likewise, Mills teaches that one of the display areas displays only information [claim 8] with video window 22, at Fig. 2, or provides access to control information [claim 9] as with control window 24, at Fig. 2, or displays an additional display element [claim 10] with video window 22, at Fig. 2.

In reference to claim 11, Mills teaches the following subject matter:

"a processor" with computer system 14, at Fig. 1, and col. 3, lines 22 - 26, 48 - 51;

"a data display screen" 18, at Fig. 1, and at col. 3, lines 23 - 24;

"cursor control device" 19, at Fig. 1, and at col. 3, line 24;

"a window generation logic" with the computer system 's software, at col. 3, line 66, to "generate and display a first window region" with view window 20, at Fig. 2, and at col. 3, line 68, wherein the "first window region comprises at least one data display area" such as with video window 22, and control window 24, at col. 3, line 68 to col. 4, line 1;

"indicia graphics generation logic" with the computer system 's software, at col. 3, line 66, to "generate user sensitive graphics for display" at col. 4, lines 32 - 40;

wherein the window generation logic determines when a data display area has been selected, signals the indicia graphics generation logic, which then initiates a response, at col. 2, lines 9 - 13.

As to claims 12 - 14, they correspond respectively to claims 6, 2, 3.

Regarding claim 15, Mills rendered anticipated the following subject matter:

"generating a first window" with view window 22, at Fig. 2, to accommodate one display area, as with control window 24, for "indicia" with control buttons 42 - 54, at Fig. 2,

Art Unit: 2415

and at col. 4, lines 37 - 40, by "executing a first programming module" with the computer system 's software at col. 3, line 66;

"displaying an indicia" at col. 4, lines 32 - 40, by executing a programming module with the computer system 's software at col. 3, line 66;

"selecting one of the indicia" at col. 4, lines 48 - 55, and sending a message for generating the display at col. 3, lines 22 - 26;

with the programming module performing a function, at col. 4, lines 38 - 40.

Mills also teaches "status information" [claim 16] such as tentative-set state, at col. 5, lines 10 - 14, or fixed-set state, at col. 5, lines 23 - 35, and describes the operating state or mode of the controller, at col. 6, lines 21 - 32.

In addition, Mills teaches "control information" [claim 17] in control window 24, at Fig. 2, and at col. 4, lines 32 - 40.

As to claim 18, Mills teaches that the first programming module requests a set of features at col. 4, lines 20 - 25, sends a message to the programming module indicative of features, and the programming module returns a message; such that the programming modules interact with each other in response to user interaction with the first programming module, all at col. 4, lines 25 - 40.

Conclusion

5. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Serial Number: 08/316,237

-6-

Art Unit: 2415

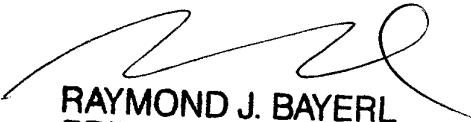
Gest et al. (U.S. patent 4,896,291) describe graphical user interface tools that include slide bars 12, dials 11, and menus 14, at Fig. 1.

Baker et al. (U.S. patent 5,428,730) teach interfaces and controls for the operation of multimedia devices that include various control panels 204, 206, 208, at Fig. 2.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Crescille N. dela Torre whose telephone number is (703) 305-9782.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3800.

Cnd
cnd
March 14, 1996


RAYMOND J. BAYERL
PRIMARY EXAMINER
ART UNIT 2415

NOTICE OF DRAFTSPERSON'S PATENT DRAWING REVIEW

PTO Draftpersons review all originally filed drawings regardless of whether they are designated as formal or informal. Additionally, patent Examiners will review the drawings for compliance with the regulations. Direct telephone inquiries concerning this review to the Drawing Review Branch, 703-305-8404.

The drawings filed (insert date) 9/30/95 are
A. not objected to by the Draftsperson under 37 CFR 1.84 or 1.152.
B. objected to by the Draftsperson under 37 CFR 1.84 or 1.152 as indicated below. The Examiner will require submission of new, corrected drawings when necessary. Corrected drawings must be submitted according to the instructions on the back of this Notice.

- 1. DRAWINGS. 37 CFR 1.84(c): Acceptable categories of drawings:
Black ink. Color.
Not black solid lines. Fig(s) _____
Color drawings are not acceptable until petition is granted. Fig(s) _____
- 2. PHOTOGRAPHS. 37 CFR 1.84(b)
Photographs are not acceptable until petition is granted. Fig(s) _____
Photographs not properly mounted (must use bristol board or photo graphic double weight paper). Fig(s) _____
Poor quality (half-tone). Fig(s) _____
- 3. GRAPHIC FORMS. 37 CFR 1.84(d)
Chemical or mathematical formula not labeled as separate figure. Fig(s) _____
Group of waveforms not presented as a single figure, using common vertical axis with time extending along horizontal axis. Fig(s) _____
Individuals waveform not identified with a separate letter designation adjacent to the vertical axis. Fig(s) _____
- 4. TYPE OF PAPER. 37 CFR 1.84(c)
Paper not flexible, strong, white, smooth, nonshiny, and durable. Sheet(s) _____
Erasures, alterations, overwritings, interlineations, cracks, creases, and fold-copy machine marks not accepted. Fig(s) _____
Mylar, velum paper is not acceptable (too thin). Fig(s) _____
- 5. SIZE OF PAPER. 37 CFR 1.84(i): Acceptable sizes:
21.6 cm. by 35.6 cm. (8 1/2 by 14 inches)
21.6 cm. by 33.1 cm. (8 1/2 by 13 inches)
21.6 cm. by 27.9 cm. (8 1/2 by 11 inches)
21.0 cm. by 29.7 cm. (DIN size A4)
All drawing sheets not the same size. Sheet(s) _____
Drawing sheet not an acceptable size. Sheet(s) _____
- 6. MARGINS. 37 CFR 1.84(g): Acceptable margins:

Paper size			
21.6 cm. X 35.6 cm. (8 1/2 X 14 inches)	21.6 cm. X 33.1 cm. (8 1/2 X 13 inches)	21.6 cm. X 27.9 cm. (8 1/2 X 11 inches)	21.0 cm. X 29.7 cm. (DIN Size A4)
T 5.1 cm. (2")	2.5 cm. (1")	2.5 cm. (1")	2.5 cm.
L .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	2.5 cm.
R .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	1.5 cm.
B .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	1.0 cm.

Margins do not conform to chart above.

Sheet(s) _____
 Top (T) _____ Left (L) _____ Right (R) _____ Bottom (B) _____

- 7. VIEWS. 37 CFR 1.84(h)
REMINDER: Specification may require revision to correspond to drawing changes.
All views not grouped together. Fig(s) _____
Views connected by projection lines or lead lines. Fig(s) _____
Partial views. 37 CFR 1.84(h) 2. _____

- View and enlarged view not labeled separately or properly. Fig(s) _____
- Sectional views. 37 CFR 1.84 (h) 3 _____
- Hatching not indicated for sectional portions of an object. Fig(s) _____
- Cross section not drawn same as view with parts in cross section with regularly spaced parallel oblique strokes. Fig(s) _____
- 8. ARRANGEMENT OF VIEWS. 37 CFR 1.84(i)
Words do not appear on a horizontal, left-to-right fashion when page is either upright or turned so that the top becomes the right side, except for graphs. Fig(s) _____
- 9. SCALE. 37 CFR 1.84(k)
Scale not large enough to show mechanism with crowding when drawing is reduced in size to two-thirds in reproduction. Fig(s) _____
Indication such as "actual size" or scale 1/2" not permitted. Fig(s) _____
- 10. CHARACTER OF LINES, NUMBERS, & LETTERS. 37 CFR 1.84(j)
 Lines, numbers & letters not uniformly thick and well defined, clean, durable, and black (except for color drawings). Fig(s) 1-10
- 11. SHADING. 37 CFR 1.84(m)
Solid black shading areas not permitted. Fig(s) _____
Shade lines, pale, rough and blurred. Fig(s) _____
- 12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1.84(p)
Numbers and reference characters not plain and legible. 37 CFR 1.84(p)(1) Fig(s) _____
Numbers and reference characters not oriented in same direction as the view. 37 CFR 1.84(p)(1) Fig(s) _____
English alphabet not used. 37 CFR 1.84(p)(2) Fig(s) _____
 Numbers, letters, and reference characters do not measure at least .32 cm. (1/8 inch) in height. 37 CFR(p)(3) Fig(s) 1, 5-10
- 13. LEAD LINES. 37 CFR 1.84(q)
Lead lines cross each other. Fig(s) _____
Lead lines missing. Fig(s) _____
- 14. NUMBERING OF SHEETS OF DRAWINGS. 37 CFR 1.84(t)
Sheets not numbered consecutively, and in Arabic numerals, beginning with number 1. Sheet(s) _____
- 15. NUMBER OF VIEWS. 37 CFR 1.84(n)
Views not numbered consecutively, and in Arabic numerals, beginning with number 1. Fig(s) _____
View numbers not preceded by the abbreviation Fig. Fig(s) _____
- 16. CORRECTIONS. 37 CFR 1.84(w)
Corrections not made from prior PTO-948. Fig(s) _____
- 17. DESIGN DRAWING. 37 CFR 1.152
Surface shading shown not appropriate. Fig(s) _____
Solid black shading not used for color contrast. Fig(s) _____

COMMENTS:

REMINDER

Drawing changes may also require changes in the specification, e.g., if Fig. 1 is changed to Fig. 1A, Fig. 1B, Fig. 1C, etc., the specification, at the Brief Description of the Drawings, must likewise be changed. Please make such changes by 37 CFR 1.312 Amendment at the time of submitting drawing changes.

INFORMATION ON HOW TO EFFECT DRAWING CHANGES

1. Correction of Informalities—37 CFR 1.85

File new drawings with the changes incorporated therein. The application number or the title of the invention, inventor's name, docket number (if any), and the name and telephone number of a person to call if the Office is unable to match the drawings to the proper application, should be placed on the back of each sheet of drawings in accordance with 37 CFR 1.84(c). Applicant may delay filing of the new drawings until receipt of the Notice of Allowability (PTOL-37). Extensions of time may be obtained under the provisions of 37 CFR 1.136. The drawing should be filed as a separate paper with a transmittal letter addressed to the Drawing Review Branch.

2. Timing of Corrections

Applicant is required to submit **acceptable** corrected drawings within the three-month shortened statutory period set in the Notice of Allowability (PTOL-37). If a correction is determined to be unacceptable by the Office, applicant must arrange to have acceptable correction resubmitted within the original three-month period to avoid the necessity of obtaining an extension of time and paying the extension fee. Therefore, applicant should file corrected drawings as soon as possible.

Failure to take corrective action within set (or extended) period will result in **ABANDONMENT** of the Application.

3. Corrections other than Informalities Noted by the Drawing Review Branch on the Form PTO 948

All changes to the drawings, other than informalities noted by the Drawing Review Branch, **MUST** be approved by the examiner before the application will be allowed. No changes will be permitted to be made, other than correction of informalities, unless the examiner has approved the proposed changes.

FORM PTO-892 (REV. 2-92)	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	SERIAL NO. 08/821004 316,237	GROUP ART UNIT 2415	ATTACHMENT TO PAPER NUMBER 5
NOTICE OF REFERENCES CITED		APPLICANT(S) CHRISTENSEN.		

U.S. PATENT DOCUMENTS

*	DOCUMENT NO.	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
A	4 8 9 6 2 9 1	1/1990	GEST ET AL.	345 375	841 156	
B	5 2 0 2 9 6 1	4/1993	MILLS ET AL.	345 375	720 159	
C	5 4 2 8 7 3 0	6/1995	BAKER ET AL.	345 375	841 740 161x	12/1992
D						
E						
F						
G						
H						
I						
J						
K						

FOREIGN PATENT DOCUMENTS

*	DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUB-CLASS	PERTINENT SHTS. DWG. PP. SPEC.
L							
M							
N							
O							
P							
Q							

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, Etc.)

R	
S	
T	
U	

EXAMINER C. DELA TORRE	DATE 14 MAR 96.
---------------------------	--------------------

* A copy of this reference is not being furnished with this office action.
(See Manual of Patent Examining Procedure, section 707.05 (a).)

- [54] **VALUATOR MENU FOR USE AS A GRAPHICAL USER INTERFACE TOOL**
- [75] **Inventors:** Stephen B. Gest; Farrell W. Wymore, both of Austin, Tex.
- [73] **Assignee:** International Business Machines Corporation, Armonk, N.Y.
- [21] **Appl. No.:** 196,922
- [22] **Filed:** May 20, 1988
- [51] **Int. Cl.⁴** G06F 3/00
- [52] **U.S. Cl.** 364/900; 340/703; 340/709; 340/724; 340/735; 364/948.2; 364/948.21; 364/300
- [58] **Field of Search** 364/200, 300, 900; 340/703, 709, 724, 735

[56] **References Cited**

U.S. PATENT DOCUMENTS

Re. 32,632	3/1988	Atkinson	340/709
Re. 32,773	10/1988	Goldwasser et al.	364/900 X
4,692,858	9/1987	Redford et al.	364/200
4,694,286	9/1987	Bergstedt	340/703
4,739,314	4/1988	McCaskill et al.	340/709
4,772,882	9/1988	Mical	340/709

OTHER PUBLICATIONS

R. M. Bateman et al., "Method for Concurrent Support

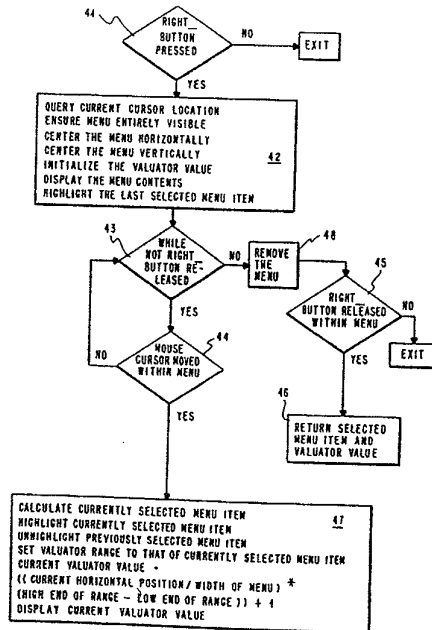
of Pop-Up Window Selection Techniques", IBM Technical Disclosure Bulletin, vol. 30, No. 10, 3/88, p. 61. L. Koved et al., "Embedded Menus: Selecting Items in Context", Communications of the ACM, 4/86, vol. 29, No. 4, pp. 312-318.

Primary Examiner—Raulfe B. Zache
Attorney, Agent, or Firm—Marilyn D. Smith

[57] **ABSTRACT**

The system and method provides a user interface tool for simultaneously selecting a menu item and a value, from a range of values, for the menu item. The user interface tool is referred to as a valuator menu, since it allows both the selection of a value from a range of values, and the selection of a menu item from a menu list. As a user moves a cursor over a menu of selectable items on a screen display, the item underneath the cursor is highlighted. In addition, as the user moves the cursor within the highlighted menu item, a value relative to the position of the cursor within that menu item is displayed. This valuator value is dynamically updated as the cursor position changes within the menu item. When the user performs an input selection, i.e. through a mouse button or a keyboard interaction, both the selected menu item and the value, relative to the cursor position, are simultaneously returned to the application program running on the data processing system.

19 Claims, 8 Drawing Sheets



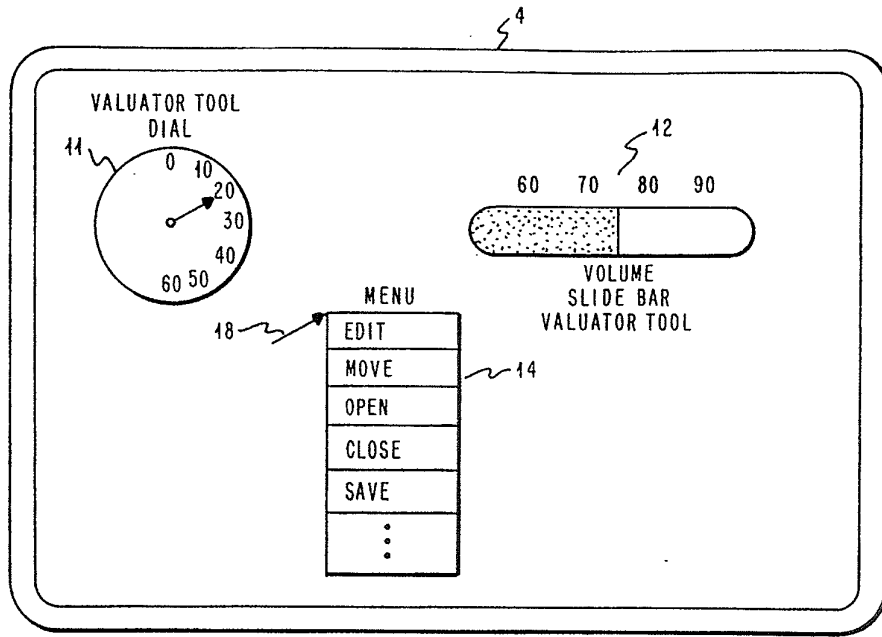


FIG. 1

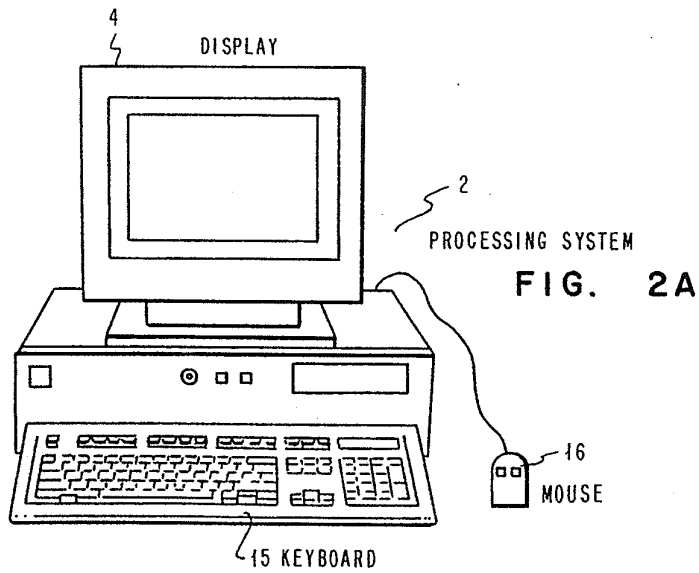


FIG. 2A

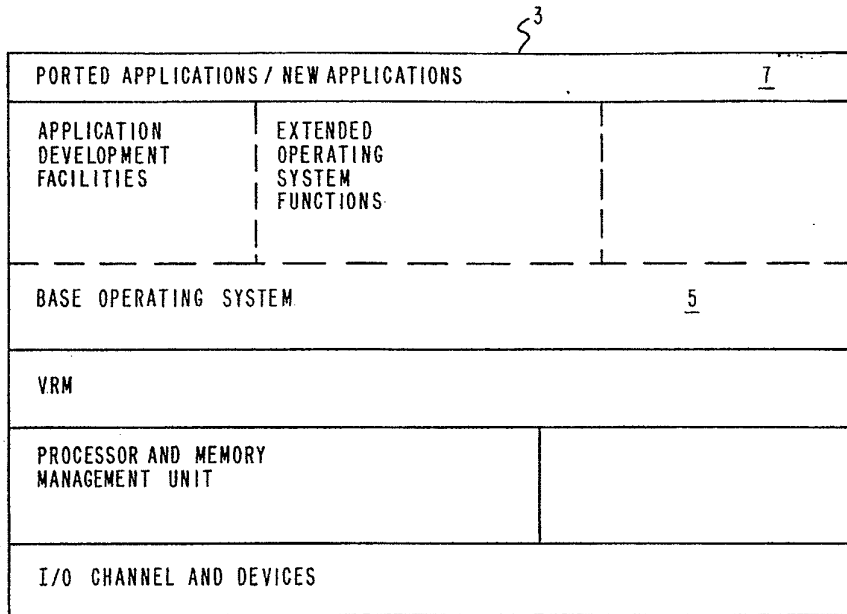


FIG. 2 B

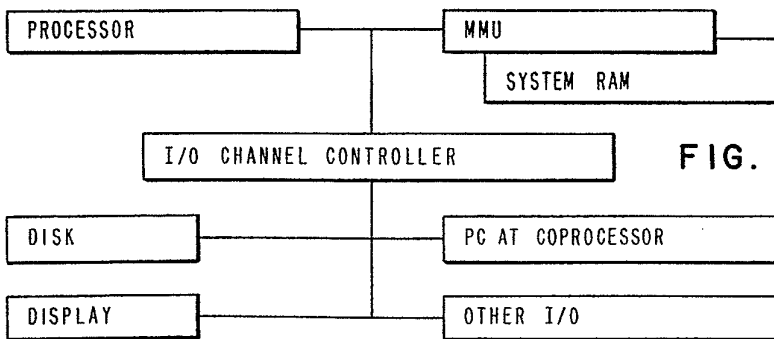
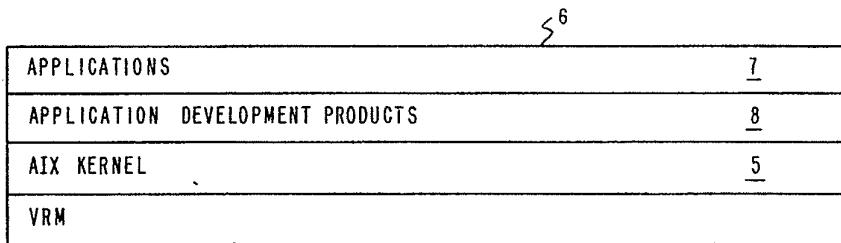


FIG. 2 C

7
4

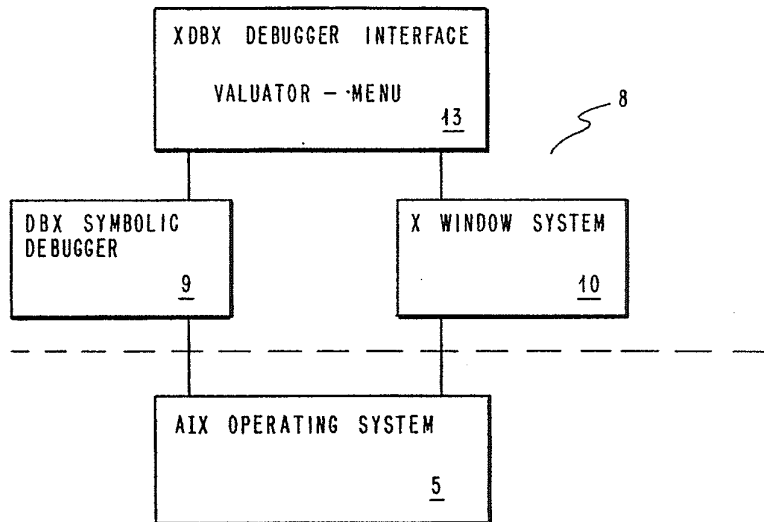


FIG. 2D

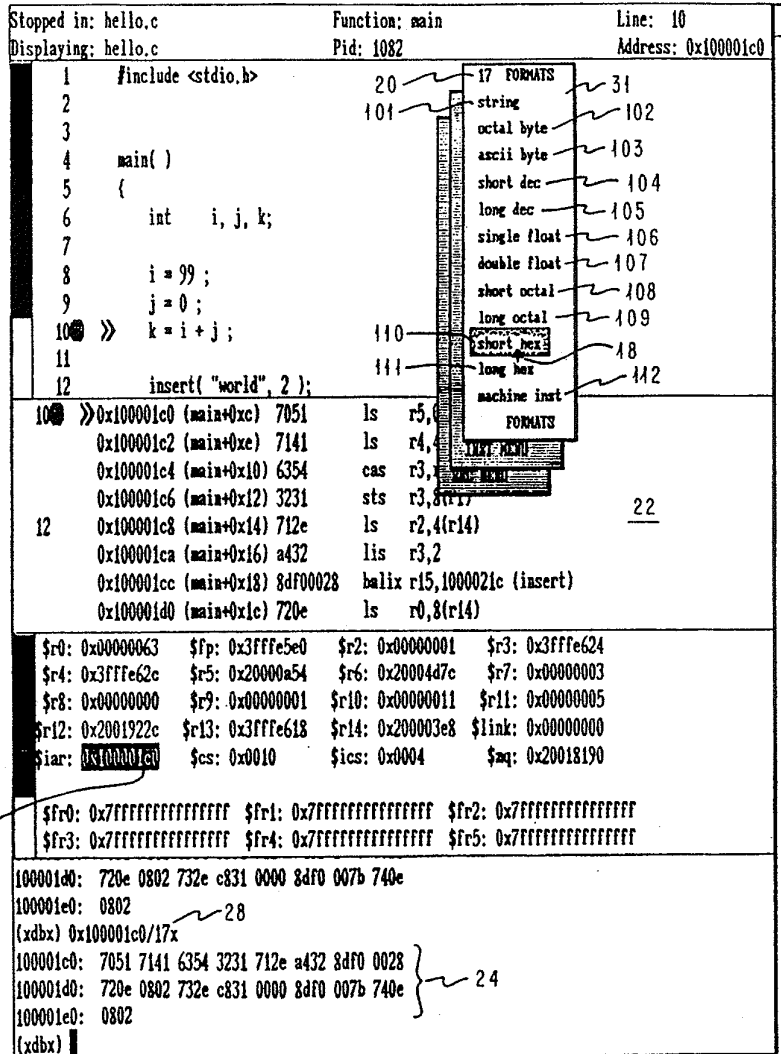


FIG. 3

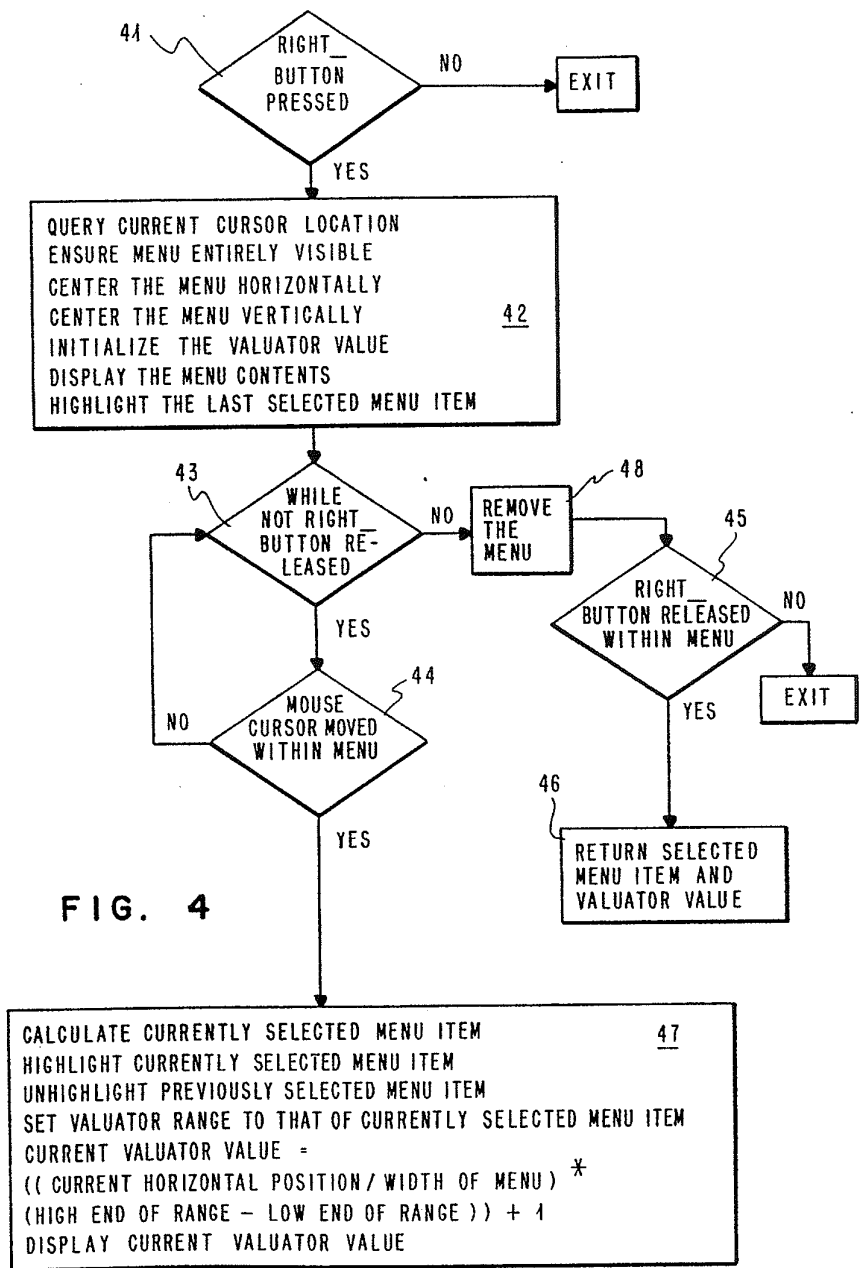


FIG. 4

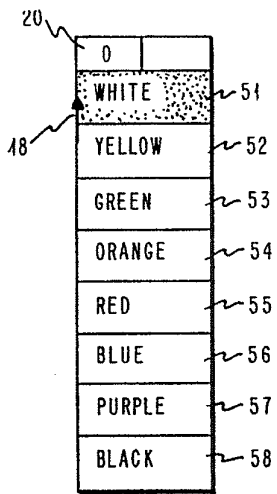


FIG. 5A

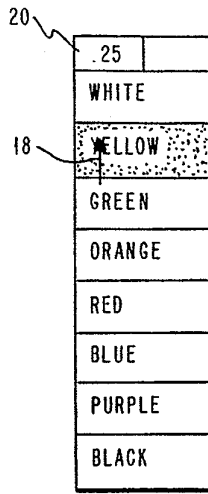


FIG. 5B

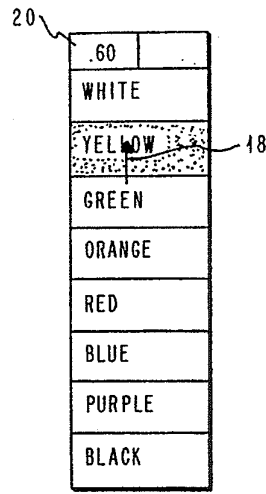


FIG. 5C

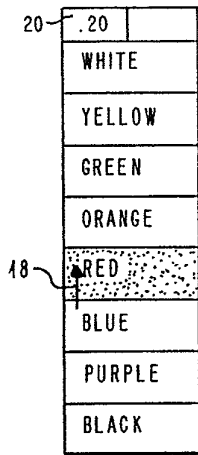


FIG. 5D

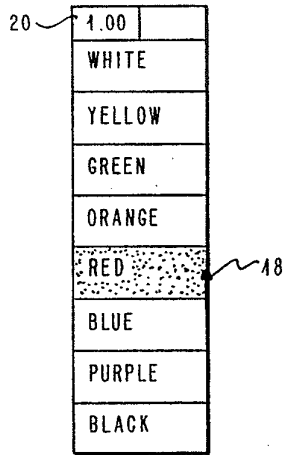


FIG. 5E

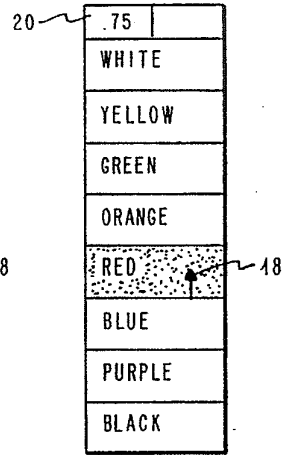


FIG. 5F

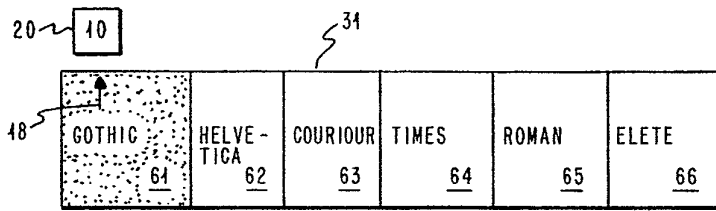


FIG. 6A

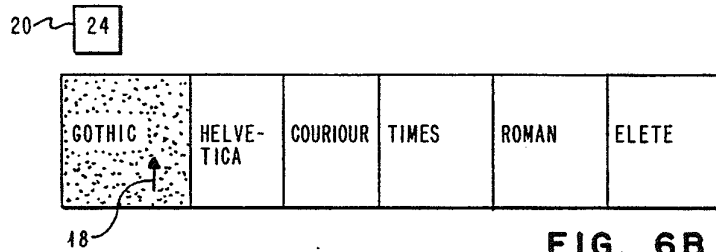


FIG. 6B

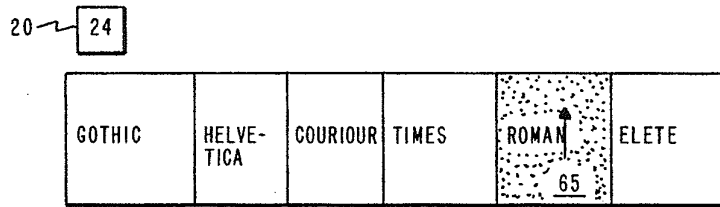


FIG. 6C

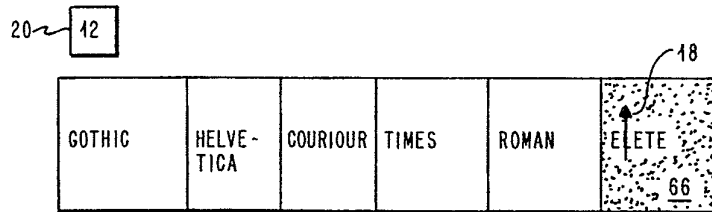


FIG. 6D

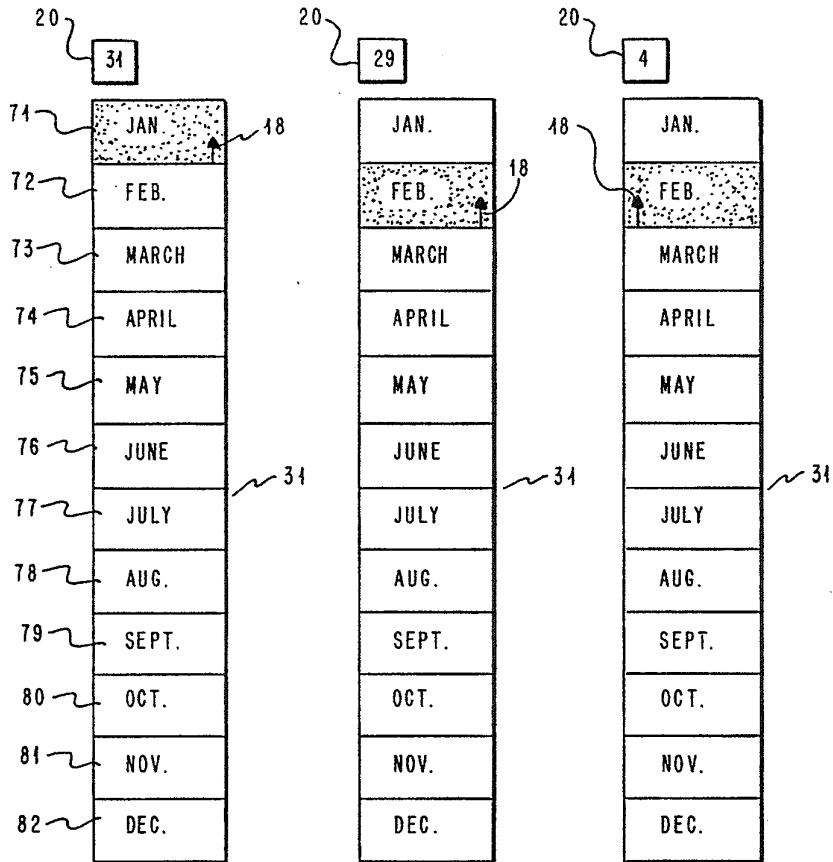


FIG. 7A

FIG. 7B

FIG. 7C

VALUATOR MENU FOR USE AS A GRAPHICAL USER INTERFACE TOOL

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

This invention relates to data processing systems having a display, and more particularly, to a user interface for selecting items from a menu displayed to a user.

DESCRIPTION OF THE RELATED ART

Windows, icons, mouse interactions, and pop-up menu systems are part of a computer user interface known in the art. A typical user interface may include a valuator tool. A valuator tool is a representation of a gauge which graphically presents a value over some range of values. As shown in FIG. 1, a dial 11 and a slide bar 12 are examples of valuator tools. The user of such a tool can graphically manipulate the valuator to indicate different values. The valuator tool returns, upon manipulation, a value with respect to its valuator type. For example, a volume control device could be represented by a slide bar 12 with graduated increments along the bar, as shown. Manipulating the slide bar returns the specified increment along the bar, thus increasing or decreasing the volume of the device.

A menu tool 14, as shown in FIG. 1, is a window containing a series of selectable items that appears when a specific mouse button(s) is pressed. The menu may appear at the current location of the mouse cursor, ("pop-up" menu), or appear below the menu's title when the user presses a mouse button(s) within that title, ("pull-down" menu).

Menu items are words or phrases that describe some type of operation that the application can perform. The user selects an item from the menu by invoking the menu, (pressing and holding down a mouse button(s)), moving the mouse cursor so that it points to the item to be selected, and releasing the mouse button(s). As the mouse cursor moves from menu item to menu item, the item currently being pointed to by the cursor is highlighted in some fashion, usually reverse video. Releasing the mouse button(s) on a highlighted item selects that item and causes the application to perform that operation.

The problem with current technology is that often it is necessary to provide parameters for operations selected from a menu. Currently, these parameters are provided by invoking some dialog between the user and the application either before or after the selection is made. Dialogs are invoked before selections are made when all the operations for a given menu possess some common subset of parameters. Users supply the requested parameters via a dialog of additional keystrokes and/or mouse actions. For a commonly performed operation, this dialog is both annoying and, time consuming for the user of the application.

SUMMARY OF THE INVENTION

An object of this invention is to economize the user's interaction in specifying multiple pieces of information to an application running on a processing system.

A further object of this invention is to reduce the number of input selections, such as through the number of keystrokes on a keyboard or through mouse movement and button selection, required by a user in a user interface.

The valuator menu addresses the above mentioned problems by providing a convenient way for the user to specify multiple pieces of information to the application with a reduced number of interactions.

The valuator menu combines the valuator tool and the menu tool to create a unique interactive tool from which the user can specify multiple pieces of information to the application program, running on the processing system, in an effective and economical manner. The user specifies this information to an application by selecting a choice from a menu, while simultaneously selecting a value, from a range of values, specific to each menu choice.

The user selects a menu item by moving the mouse cursor vertically over the menu in the manner described above. The user can additionally select a valuator value, from a range of values, by moving the cursor horizontally over the menu. The current valuator value is displayed within the menu, and is dynamically updated with respect to the mouse cursor's horizontal position within the menu item. The valuator value may be in any range, and adjusted to any appropriate scale.

Releasing the mouse button(s) over a highlighted menu item returns multiple pieces of information to the application via a single mouse operation. The information returned consists of the selected menu operation to be performed, and a valuator value pertaining to that operation.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 illustrates a dial valuator tool, a slide bar valuator tool, and a pop-up menu, all known in the art as graphical user interface tools.

FIG. 2A illustrates the hardware including a display of a processing system for utilizing this invention.

FIG. 2B illustrates the logical structure of the processing system of the preferred embodiment.

FIG. 2C illustrates the physical structure of the processing system of the preferred embodiment.

FIG. 2D illustrates the software components of the preferred embodiment.

FIG. 3 illustrates a screen display output of the valuator menu in a preferred embodiment.

FIG. 4 illustrates a flow chart for implementing the valuator menu of the present invention.

FIGS. 5A-5F illustrates various screen displays showing the function of the valuator menu in simultaneously selecting a menu item and a value of the menu item in another embodiment of the invention.

FIGS. 6A-6D illustrates various screen displays showing the function of the valuator menu in simultaneously selecting a menu item and a value of the menu item in another embodiment of the invention.

FIGS. 7A-7C illustrates various screen displays showing the function of the valuator menu in simultaneously selecting a menu item and a value of the menu item in another embodiment of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The preferred embodiment of the present invention was targeted for the IBM RT PC 2, FIGS. 2A, 2B, 2C, 5 running the AIX operating system 5. The valuator menu of the present invention could be utilized with display 4, FIG. 2A provided with the processing system 2. FIG. 2B shows the logical structure 3 of the processing system 2, FIG. 2A, FIG. 2C shows the physical structure 6 of the processing system 2, FIG. 2A. 10

For more information on the RT PC, and the AIX operating system, the following references are herein incorporated by reference. Bach, M. J., *The Design of the UNIX Operating System*, Prentice Hall, 1986. Lang, T. G. and Mothersole, T. L., *Design of the RTPC VRM Nucleus*, Sept. 1, 1986. *AIX Operating System Commands Reference*, IBM Corporation, SC23-0790. *AIX Operating System Managing the AIX Operating System*, IBM Corporation, SC23-0793. *AIX Operating System Programming Tools and Interfaces*, IBM Corporation, SC23-0789. *AIX Operating System Technical Reference*, Volumes 1 and 2, IBM Corporation, SC23-0808 and SC23-0809. *IBM RT Personal Computer Technology*, IBM Corporation, SA23-1057, 1986. *Virtual Resource Manager Technical Reference*, Volumes 1 and 2, IBM Corporation, SC23-0816 and SC23-0817. 15

The valuator menu user interface tool of this invention is a vehicle with which a user can specify several pieces of information for a commonly used operation with a minimal amount interaction. 20

The present invention was first implemented in an embodiment comprising an application called "Xdbx" 13 FIG. 2D that presents a modern interface to the "dbx" symbolic debugger 9. This interface improves the general presentation and usability of the dbx symbolic debugger. The dbx and Xdbx symbolic debuggers are described in the *IBM RT PC Programming Tools and Interfaces*, Version 2.2, IBM Corporation, which is herein incorporated by reference. The features of this invention are a part of the *IBM RT PC Advanced Interactive Operating System Extended Services Program*, which is herein incorporated by reference. 25

As shown in FIG. 2D, the Xdbx debugger interface 13 and the dbx symbolic debugger 9, along with X-Windows 10, are application development products 8 as shown in FIG. 2C. 30

The interface 13 makes use of the X-Windows system 10, which supplies most of the primitive window management tools that allow more complex tools, like the valuator menu, to be built. More information on X-Windows can be found in *IBM RT PC X-Windows Version 1.1, X-Windows User Guide & Reference*, September 1987, which is herein incorporated by reference. The Xdbx debugger 13 allows the user to specify each dbx operation 9, giving the user complete functionality, without being constrained by each operation's syntax. 35

One commonly used operation in the dbx symbolic debugger 9 is the ability to list assembly instructions 22, FIG. 3, or display the contents of an address 26 as shown by contents 24. These operations require three pieces of information: an address 26, the number of memory items to display 20, and a mode specifying how to display the memory 110. A shown in FIG. 3, Dbx supports the following display modes: string 101, octal byte 102, ascii byte 103, short decimal 104, long decimal 105, single precision float 106, double precision float 40

107, short octal 108, long octal 109, short hexadecimal 110, long hexadecimal 111, and machine instruction 112.

With reference to both FIG. 3 and FIG. 4, the valuator menu tool allows the user to specify the above information to Xdbx in an economical and effective manner. The user selects the starting address 26 of the range of memory to display by moving the mouse cursor 18 so that it points to that address 26 and clicking the left mouse button 16, FIG. 2A. The selected address 26 would then be highlighted in reverse video as shown. 45

Having selected the starting address 26, the user could invoke the valuator menu by pressing and holding down the right mouse button, step 41 FIG. 4. As shown in FIG. 3, the user moved the cursor to another location on the screen to enhance clarity of FIG. 3 before invoking the valuator menu. However, this is not necessary, as the valuator menu 31 could appear at location 26. 50

Nevertheless, the valuator menu 31 appears centered horizontally about the current mouse cursor location, step 42. The valuator menu 31 is positioned vertically such that the cursor appears over the same menu item 110 that was previously selected, or over the first menu item 101, if it is the first time that the valuator menu 31 is invoked. 55

The user can simultaneously select a menu item, i.e. a display mode 101-112, and one of a number of values, i.e. number of memory items 20, to be displayed. The user could select a display mode 101-112 by moving the mouse cursor vertically over the valuator menu 31. As the mouse cursor moves from menu item to menu item, the item 110 currently being pointed to by the cursor 18 is highlighted in reverse video, step 42, FIG. 4. 60

Simultaneously, the user could select one of a number of values, i.e. memory items 20, to be displayed by moving the cursor horizontally over the valuator menu 31, step 44. As the user moves the cursor horizontally within the menu 31, the valuator value 20 is updated with respect to the cursor's horizontal position, step 47. Since the menu is initially centered about the mouse cursor, the initial value represents a mid range value of the range specified. 65

The menu items 101-112 would consist of the possible memory display modes, while the valuator value 20 would represent the current number of memory items to display. Releasing the right button over a highlighted display mode, step 45, returns both that display mode 110, and the number of memory items to display 20, back to the Xdbx program via one single mouse button release, step 46. With this information, Xdbx could now invoke the dbx operation to display a memory range 24, preserving the dbx syntax as shown by output 28, FIG. 3. The dbx syntax "0x100001c0/17x" means that beginning at memory location 0x100001c, seventeen 16 bit integers, i.e. short integers, will be displayed as hexadecimal numerals. These seventeen integers are shown as numeral 24 in FIG. 3. 70

For this commonly used operation, the valuator menu was the perfect vehicle to allow the user to specify multiple pieces of information in an efficient and economical fashion.

The following program design language code illustrates the above operation:

```

if ( right_button pressed ) {
  query the current mouse cursor location;
  adjust the menu such that it is entirely visible;
  center the menu horizontally about the current
  mouse cursor location;
}

```

-continued

```

center the menu vertically about the last selected
memory display mode;
set the valuator_value to .5 * valuator_range;
display the menu title, menu items, and valua-
tor_value;
highlight the last selected memory display mode in
reverse video;
while ( not right_button released ) {
if ( mouse cursor moved within menu ) {
calculate currently selected memory
display mode;
highlight currently selected memory
display mode;
unhighlight previously highlighted memory
display mode;
calculate the current valuator_value =
(( current horizontal position /
width of menu ) * ( upper
range - lower range )
+ 1;
display current valuator_value represent-
ing number of memory items to
display;
}
}
/* right_button released */
remove the menu;
if ( right_button released within menu )
display the number of memory items indicated
by the valuator_value in the selected
memory mode;
}

```

Copyright IBM Corporation 1988

The program design language code listed above is specific to this preferred embodiment. However, the valuator menu of the present invention is not limited to the specific application as described in reference to the previous embodiment employing the Xdbx debugger. For a more general embodiment, references to memory items, display modes, and memory modes would be replaced by other words representing a different embodiment. Generally, for other embodiments, the selected action will be implemented with the valuator value as its argument.

The valuator menu technique can be implemented in a host of other applications. For example, this tool can be used to specify a certain hue of a particular color. FIG. 5A through FIG. 5F illustrate sequences of a menu as a cursor is moved to simultaneously specify a certain hue of a particular color. The valuator menu items 51-58 could consist of the range of possible colors starting with white 51 and ending with black 58. The valuator value 20 would represent the amount of saturation for the color currently being pointed to by the mouse cursor, which is highlighted in reverse video. For this example, the valuator value 20 might range from 0 to 1 in increments of hundredths, indicating the saturation of the color. Moving vertically over the menu would select a color, while moving horizontally within the menu would change the amount of saturation for that particular color.

As shown in FIG. 5A, the cursor 18 is pointing to color white 51 at the farthest left position indicating a zero saturation level as shown by the valuator value 20. In FIG. 5B the cursor is moved down vertically to the yellow color item 52. As the cursor 18 is also moved horizontally to the right, the valuator value 20 changes dynamically with the horizontal position indicating a change in the value of the saturation level that may be selected by the user. FIG. 5C illustrates the dynamic change in the valuator value 20 as the cursor 18 location changes its relative horizontal position. FIGS. 5D, 5E,

and 5F also illustrate this dynamic change in the valuator value 20 as the cursor 18 location changes its relative horizontal position.

In this manner, the user could select simultaneously a color item 51-58 and one of a plurality of saturation levels for the color item, in one click of the mouse button. The following program design language code further illustrates the simultaneous selection of two items with one user input action.

Copyright IBM Corporation 1988

```

if ( right_button pressed ) {
query the current mouse cursor location;
adjust the menu such that it is entirely visible;
center the menu horizontally about the current
mouse cursor location;
center the menu vertically about the last selected
color;
set the saturation value to .5 * saturation range;
display the menu title, menu color items, and
saturation value;
highlight the last selected color item in reverse
video;
while ( not right_button released ) {
if ( mouse cursor moved within menu ) {
calculate currently selected color item;
highlight currently selected color item;
unhighlight previously highlighted color
item;
calculate the current saturation value =
(( current horizontal position /
width of menu ) * saturation
range ) + 1;
display current saturation value;
}
}
/* right_button released */
remove the menu;
if ( right_button released within menu )
return selected color and saturation value
to the application;
}

```

Another embodiment of the valuator menu is as a user interface tool that allows the user to simultaneously select a font style and a character pitch for that font. FIG. 6A through FIG. 6D illustrates a possible sequence of a visual representation on a display as the cursor 18 is moved throughout the menu 31. FIGS. 6A-6D also illustrate that the menu items 61-66 can be listed horizontally instead of vertically as previously shown. Also, the selection of the pitch, i.e. the valuator value, could also occur relative to the same horizontal direction as the menu items 61-66. As the cursor 18 is moved into a new menu item, the valuator value 20 would reset for the new menu item.

In this embodiment, as shown in FIGS. 6A-6D, the valuator menu items 61-66 would represent the possible font styles, while the valuator value 20 would represent the possible character pitches for each font style. This range of character pitches could be different for each font. Moving over the menu 31 would select a different font style, while simultaneously changing the range of the valuator values 20, i.e. character pitches, associated with the selected font style. Moving the cursor within the menu item would update the value with the possible character pitches for that particular font style.

As shown in FIG. 6A, the cursor location indicates menu item 61 in reverse video, with a character pitch of 10, as indicated by the valuator value 20. As the cursor 18 position is moved horizontally, (the vertical positioning of the cursor 18 is not relevant in this embodiment)

the character pitch changes to 24 as indicated by valuator value 20 in FIG. 6B. As shown in FIG. 6C, the Roman font style 65 is indicated in reverse video by the cursor 18 position, along with the 24 character pitch as indicated by valuator value 20. As the cursor 18 position is moved horizontally to the right as shown in FIG. 6C, the valuator value resets relative to the next menu item 66. The relative cursor 18 position within menu item 66. FIG. 6D, indicates simultaneously the menu item 66 is to be selected along with a character pitch of 12.

The following program design language code illustrates this embodiment as shown in FIGS. 6A-6D.

```

if ( right_button pressed ) {
  query the current mouse cursor location;
  adjust the menu such that it is entirely visible;
  center the menu horizontally about the current
  mouse cursor location;
  center the menu vertically about the last selected
  font style item;
  set the character pitch value to .5 * pitch range
  of last selected font style item;
  display the menu title, font style items, and
  character pitch value;
  highlight the last selected font style item in
  reverse video;
  while ( not right_button released ) {
    if ( mouse cursor moved within menu ) {
      calculate currently selected font style
      item;
      highlight currently selected font style
      item;
      unhighlight previously highlighted font
      style item;
      set character pitch range to that of the
      currently selected font style item;
      calculate the current character pitch
      value =
      (( current horizontal position /
      width of menu ) * pitch range ) +
      1;
      map character pitch value to pitch value
      range
      display current mapped character pitch
      value;
    }
  }
  /* right_button released */
  remove the menu;
  if ( right_button released within menu )
  return selected font style and character
  pitch value to the application;
}

```

Copyright IBM Corporation 1988

A further embodiment of the present invention employs a calendar menu as shown in FIGS. 7A-7C. With the user interface of the present invention, a user could check the plans for a particular day of the year in an economical manner. The valuator menu items 71-82 would represent the months of the year, while the valuator value 20 would represent the days for each month. The range of days per month would obviously change per month, as shown in FIG. 7A and FIG. 7B with respect to the same relative position of the cursor 18 in the menu item, and the different resulting valuator value 20.

As shown in FIGS. 7A-7C, moving the cursor 18 vertically over the menu 31 would select a menu item 71-82 in reverse video. The range of the valuator value 20 would change relative to the highlighted menu item. Moving the cursor 18 horizontally within the menu 31 would update the value 20 displayed representing the day for the particular month highlighted. With this information, an application could display the user's

agenda for that particular day of the year. The user simultaneously selected a menu item (month), and a value (a particular day) of that selected menu item with one user input selection.

The following program design language code illustrates the preferred embodiment as previously described.

```

if ( right_button pressed ) {
  query the current mouse cursor location;
  adjust the menu such that it is entirely visible;
  center the menu horizontally about the current
  mouse cursor location;
  center the menu vertically about the last selected
  month item;
  set the day of the month value to .5 * range of
  days of last selected month item;
  display the menu title, month items, and day
  value;
  highlight the last selected month item in reverse
  video;
  while ( not right_button released ) {
    if ( mouse cursor moved within menu ) {
      calculate currently selected month item;
      highlight currently selected month item;
      unhighlight previously highlighted month
      item;
      set range of days to that of the currently
      selected month item;
      calculate the current day value =
      (( current horizontal position /
      width of menu ) * range of days )
      + 1;
      display current day value;
    }
  }
  /* right_button released */
  remove the menu;
  if ( right_button released within menu )
  return selected month and day value to the
  application;
}

```

Copyright IBM Corporation 1988

While the invention has been particularly shown and described with reference to a preferred embodiment and other embodiments, it will be understood by those skilled in the art that various changes in form and detail may be made without departing from the spirit and scope of the invention. For example, various changes may include, but are not limited to, the interchangeability of references to either a horizontal direction or a vertical direction. In addition, a keyboard 15 (FIG. 2A) interaction or other input device can be used instead of using the mouse 16 input device as described herein. Also, the additional embodiments were shown as examples of the various uses of the present invention. It would be within the scope of this invention to implement the valuator menu of this invention in other embodiments not specifically discussed herein.

We claim:

1. A user interface for a data processing system, comprising:
 - means for displaying a list having a plurality of user actions; and means, coupled to said means for displaying, for simultaneously selecting one of said actions and a value within a range of values of said action.
2. A system for selecting an action in a data processing system, comprising:
 - means for displaying a list of items for a plurality of actions;

9

means, coupled to said means for displaying, for selecting one of said plurality of actions in response to a movement of a cursor in a first direction; and means, coupled to said list, for selecting a value of said action in response to a second movement of the cursor in a second direction within said item. 5

3. The system of claim 2 wherein the means for selecting a value further comprises means for displaying a dynamically updated indicator representing the value corresponding to a location of said cursor in said second direction. 10

4. A user interface for use with a screen display in a data processing system, said interface comprising: means for moving a cursor in a first direction over a menu displaying a plurality of items; 15

means, coupled to said items, for correlating a range of values of each of said items relative to a position of said cursor within each one of said items of said menu;

means, coupled to said means for moving a cursor, for displaying a correlated value for said cursor position; and 20

means, coupled to said means for displaying and said means for moving a cursor, for dynamically updating a displayed value as said cursor position changes. 25

5. The interface of claim 4 further comprising means, coupled to said menu, for simultaneously specifying, to an application running on said processing system, a selected item and the correlated value of said selected item, relative to the position of said cursor, by a single user input interaction. 30

6. A user interface for use with a screen display in a data processing system, said interface comprising:

means for displaying a menu having a plurality of color items; 35

means, coupled to said means for displaying, for positioning a cursor over at least one of said color items;

means, coupled to said color items, for correlating a saturation value within a range of a plurality of saturation values for each of said plurality of color items to a cursor position within at least one of said color items; and 40

means, coupled to said menu, for simultaneously selecting one of said plurality of color items and said saturation value of said color item with a single input interaction. 45

7. A user interface for use with a screen display in a data processing system, said interface comprising: 50

means for displaying a menu having a plurality of font style items;

means, coupled to said means for displaying, for positioning a cursor over at least one of said font style items; 55

means, coupled to said font style items, for correlating a character pitch value within a range of a plurality of character pitch values for each of said plurality of font style items to a cursor position within said font style item; and means, coupled to said menu, for simultaneously selecting one of said plurality of font style items and said character pitch value of said font style item with a single input interaction. 60

8. A user interface for use with a screen display in a data processing system, said interface comprising:

means for displaying a menu having a plurality of items;

10

means, coupled to said means for displaying, for positioning a cursor over at least one of said items;

means, coupled to said items, for correlating a value within a range of a plurality of values for each of said plurality of items to a cursor position within said item; and

means, coupled to said menu, for simultaneously selecting one of said plurality of items and said value of said item with a single input interaction.

9. A method for selecting an action in a data processing system, said method comprising:

displaying a menu having a plurality of user actions; positioning a cursor within one of said actions of said menu; selecting simultaneously said action and a value within a range of a plurality of values of said action according to a position of the cursor within said action. .

10. A method for selecting an action in a data processing system, said method comprising:

displaying means having a plurality of actions; moving a cursor within said menu for selecting one of said plurality of actions; and moving a cursor within said action for selecting a value from a plurality of values of said action. 5

11. The method of claim 10 wherein the step of moving the cursor within said action further comprises the step of displaying a dynamically updated indicator representing the value of said action relative to a location of said cursor.

12. A method for selecting an action in a data processing system, said method comprising:

moving a cursor in a first direction over a menu displaying a plurality of items; correlating a range of values of each of said items relative to a position of said cursor within each one of said items of said menu; displaying a correlated value for said cursor position; and updating dynamically a displayed value as said cursor position changes. 10

13. The method of claim 12 further comprising the step of running simultaneously, to an application running on said processing system, a selected item and a correlated value of said selected item, relative to the position of said cursor, by a single user input interaction.

14. A method for selecting an action in a data processing system, said method comprising:

displaying a menu of a plurality of items; positioning a cursor over at least one of said items; correlating a value within a range of a plurality of values for each of said plurality of items to a cursor position within said item; and selecting simultaneously one of said plurality of items and said value of said item with a single input interaction. 15

15. A method for selecting an action in a data processing system, said method comprising:

displaying a list having a plurality of user actions; and selecting, simultaneously, one of said actions and a value within a range of values of said action.

16. A computer program comprising: first instruction means for causing a display of a list having a plurality of user actions; and

second instruction means, coupled to said first instruction means, for causing a simultaneous selection of one of said actions and a value within a

11

range of values of said action in response to a single user interaction.

17. A computer program comprising:

first instruction means for causing a display of a menu having a plurality of items;

second instruction means, coupled to said first instruction means, for causing a correlation of a value within a range of a plurality of values for each of said plurality of items to a cursor position within said item; and

third instruction means, coupled to said second instruction means, for causing a simultaneous selection of one of said plurality of items and said value of said item in response to a single input interaction.

12

18. A computer program of claim 16 further comprising fourth instruction means, coupled to said second instruction means, for causing a display of the value correlated to said cursor position; and fifth instruction means, coupled to said fourth instruction means, for causing a dynamic update of the value displayed as said cursor position changes.

19. A computer program comprising:

first instruction means for causing a display of a menu having a plurality of user actions;

second instruction means, coupled to said first instruction means, for causing a simultaneous selection of one of said actions and a value within a range of a plurality of values of said action according to a position of a cursor within said action.

* * * * *

20

25

30

35

40

45

50

55

60

65



US005202961A

United States Patent [19]

[11] Patent Number: 5,202,961

Mills et al.

[45] Date of Patent: Apr. 13, 1993

[54] SEQUENTIAL INFORMATION CONTROLLER

[75] Inventors: Michael Mills; Jonathan Cohen, both of San Francisco; Yin Y. Wong, Cupertino, all of Calif.; Ian S. Small, Toronto, Canada

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

[21] Appl. No.: 536,913

[22] Filed: Jun. 8, 1990

[51] Int. Cl.⁵ G06F 15/40

[52] U.S. Cl. 395/159; 395/161; 395/155; 340/747

[58] Field of Search 364/518, 519, 521, 522; 340/747, 750; 395/155, 159, 161

[56] References Cited

U.S. PATENT DOCUMENTS

4,715,008	12/1987	Jones	364/563
4,789,962	12/1988	Berry et al.	364/900
4,890,249	12/1989	Yen	364/578
4,922,238	5/1990	Aoki et al.	340/726
4,933,881	6/1990	Schmidt	364/525
4,954,966	9/1990	Mooney et al.	364/518

Primary Examiner—Gary V. Harkcom

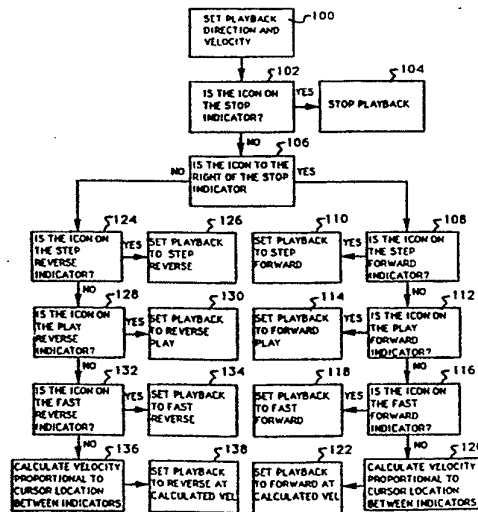
Assistant Examiner—Phu K. Nguyen

[57] ABSTRACT

A sequential information controller for use in combination with a computer comprising a slider bar, integrating standard velocity and direction indicators, and a control icon. The user controls the playback velocity and direction of the sequential information, such as video information, on the display of the computer through an interactive user interface, which works in combination with the computer's system software. The

interface includes an interactive slider bar and an interactive control icon which are depicted within an operating window of the display. The slider bar includes both playback direction/velocity indicators and control buttons. The playback of video information is primarily controlled by manipulating the image of the control icon with respect to the image of the slider bar. The image of the control icon is that of a human-like hand, which holds the slider bar image in either its open or closed hand, depending on the desired playback mode, and can be moved along the length of the slider bar image by a control device, such as a mouse, to control the playback direction and velocity of the video information on the display. When the control icon's hand is open, the control icon is spring loaded from the stop playback position of the slider bar. To keep the control icon from automatically returning to the stop playback position when the control icon's hand is open, the user must continue to hold the control icon in position with the mouse. When the control icon's hand is closed, the control icon will remain at any position along the slider bar it is placed, until moved by operation of the mouse, or until returned to the spring-loaded mode and released by the mouse, thereby allowing it to spring back to the stop position. The mode of the control icon is modulated by positioning the cursor over the control icon, selecting the control icon, and moving the cursor either up or down. Moving the cursor down, when the hand is open, closes the hand and freezes the playback velocity at the speed selected when the hand was closed. Moving the cursor up, when the hand is closed, opens the hand and spring loads the control icon. The control icon can also be rapidly moved to different positions by simply clicking on the slide bar with the mouse, regardless of the mode of operation.

45 Claims, 5 Drawing Sheets



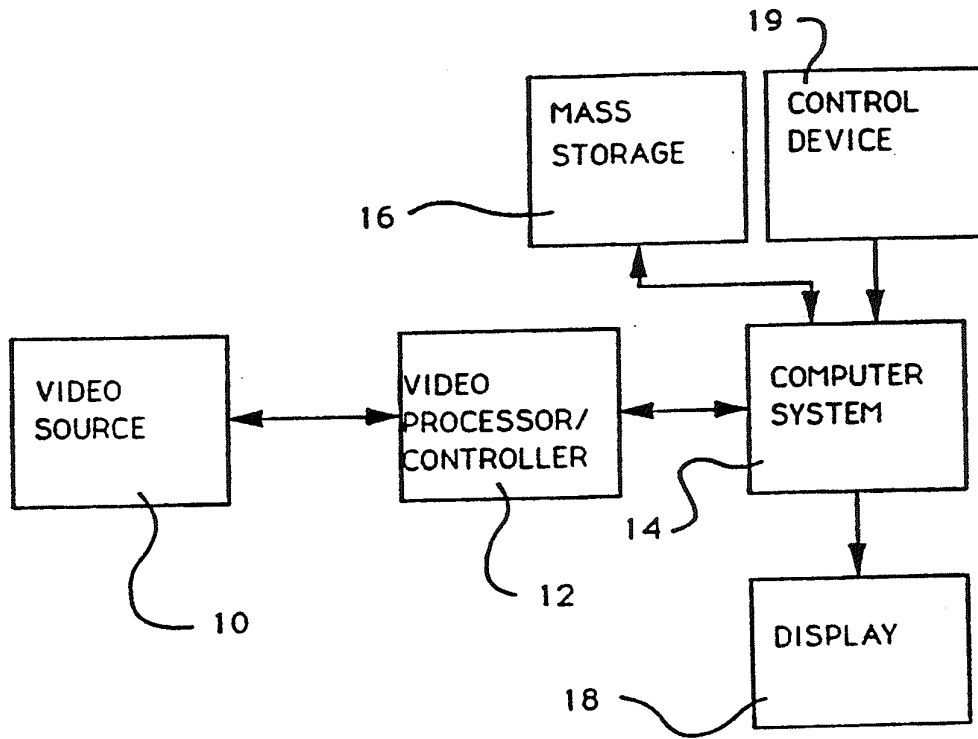


FIG. 1

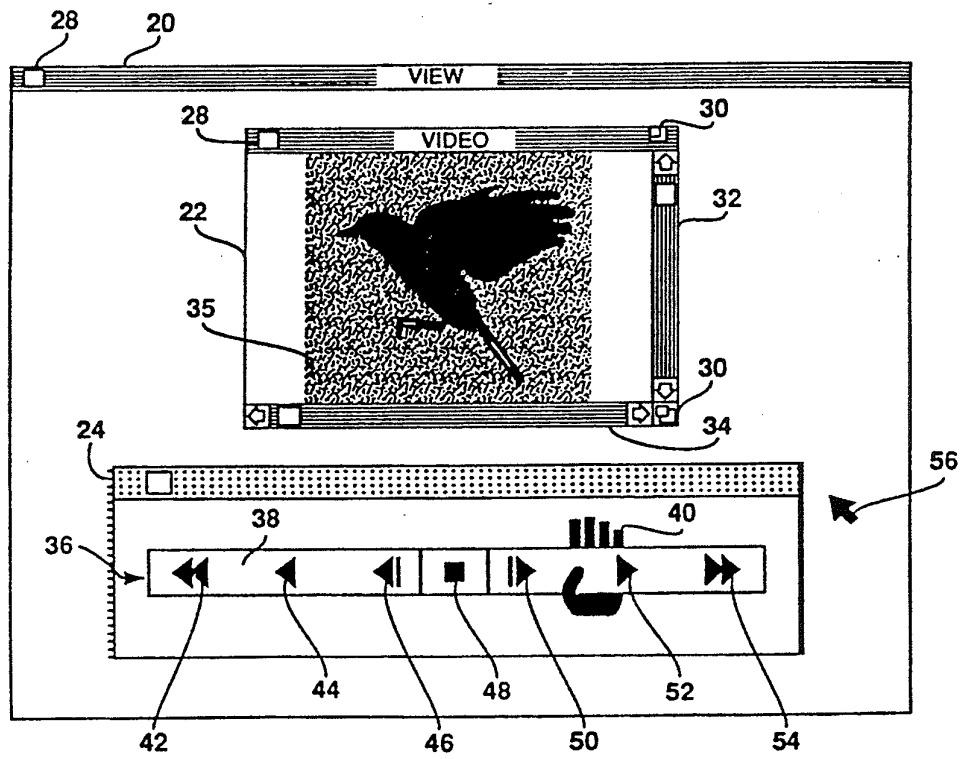


FIG. 2

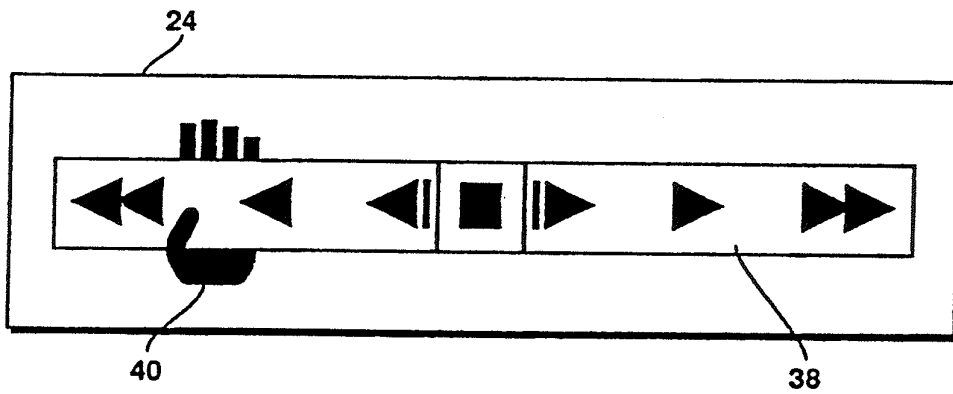


FIG. 3

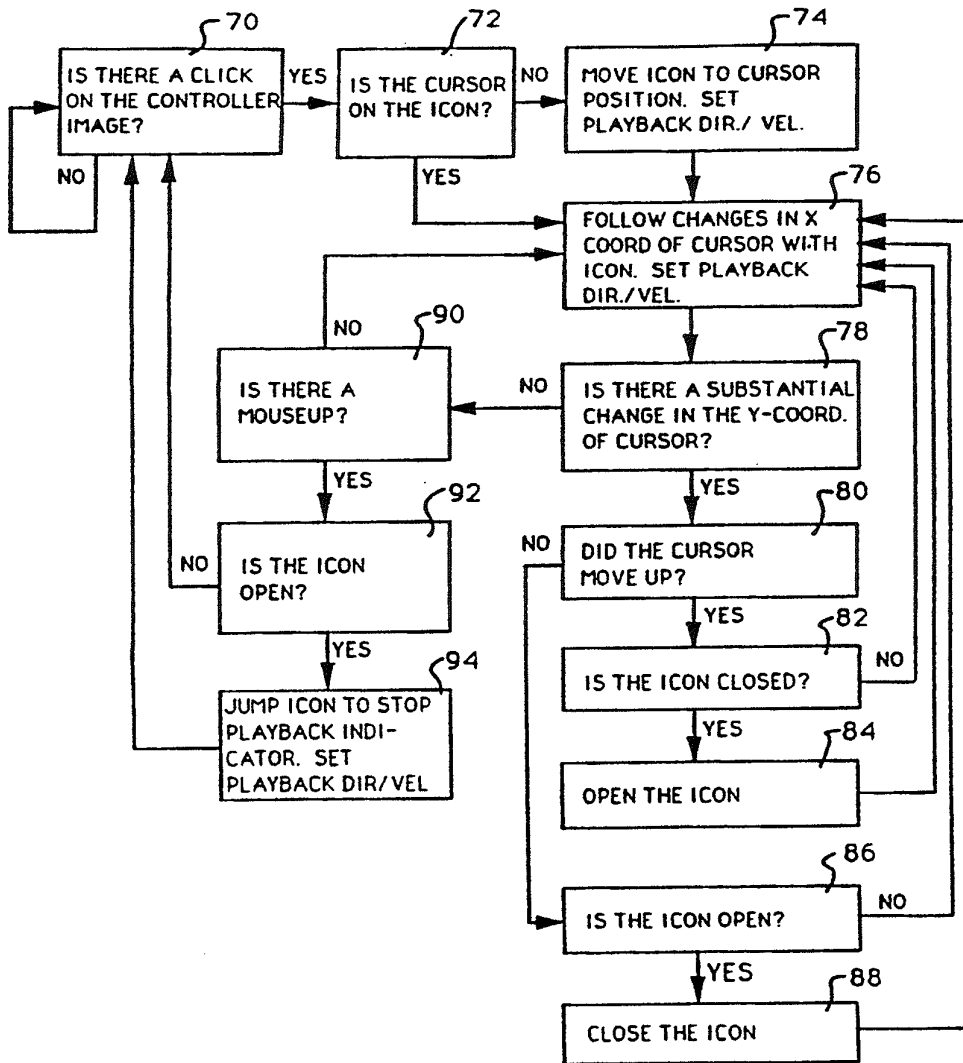


FIG. 4

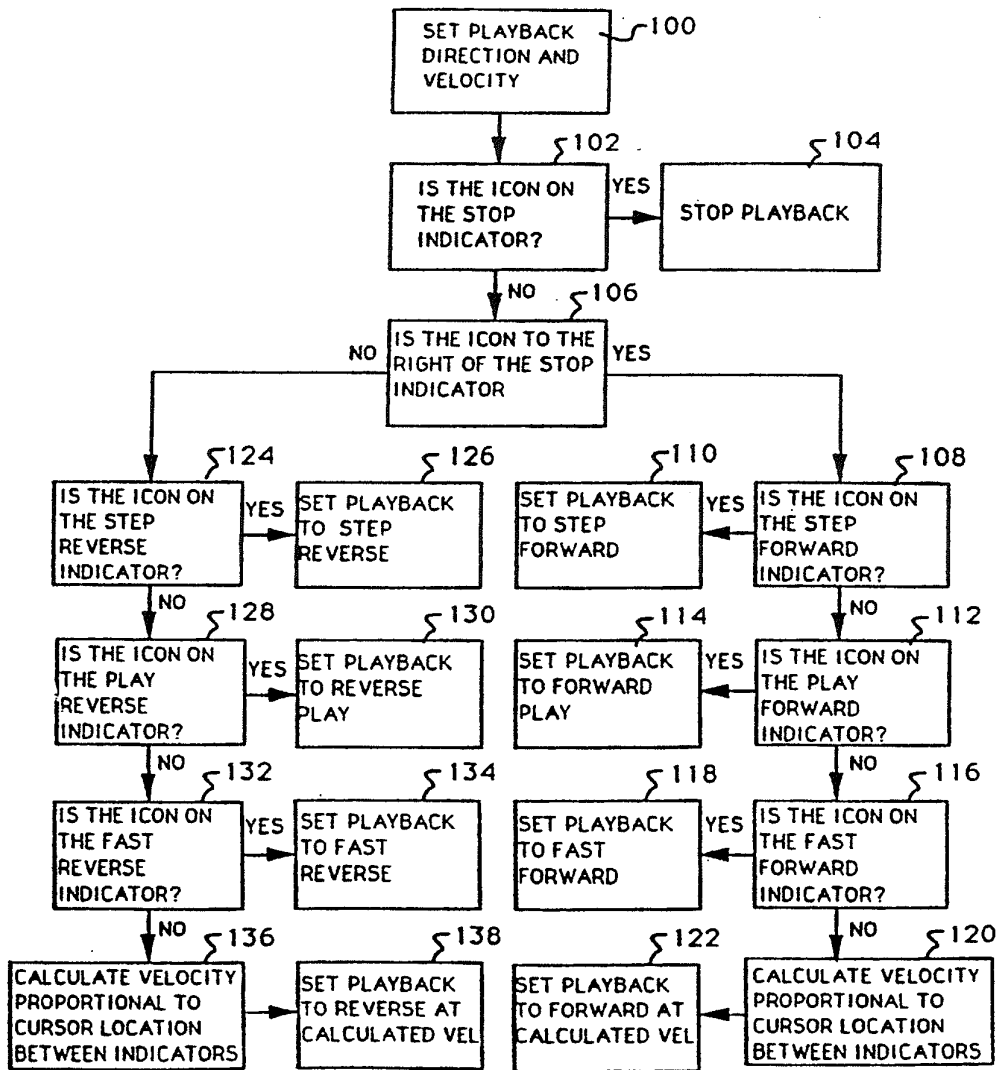


FIG. 5

SEQUENTIAL INFORMATION CONTROLLER

FIELD OF THE INVENTION

The present invention relates generally to computerized sequential information controllers, such as video playback and editing devices, and more particularly to a novel method and apparatus for controlling the playback direction and velocity of any type of sequential information, such as portions of video information.

BRIEF DESCRIPTION OF PRIOR ART

Many computerized sequential information or data controllers, such as video playback and editing systems, allow a user to preview large chunks of analog or digital audio and/or video information stored in one or more different types of devices, such as VCRs, optical disc players, and audio players. A user typically regulates the direction and velocity at which the video information is displayed through use of software or hardware controls, such as a "jog-shuttle" potentiometer (a device which causes the forward or backward display velocity of the video information to increase or decrease depending on the particular direction, and how far, the jog-shuttle's control knob is turned).

In recent years, many of these computerized video viewing and/or editing systems have replaced manual controls like the jog-shuttle with software-based tools that allow the user to regulate the same functions through operation of a cursor control device, such as a mouse. An example of a prior art software tool is a one-state slider tool developed at the Multimedia Laboratory of the Massachusetts Institute of Technology. The MIT slider is depicted together with the video information on the same display screen of a computer when in use. The slider tool consists of an elongated slider bar image having a small spring-loaded control line image which can be grabbed by a user with a mouse and dragged in either a forward or reverse play direction along the slider bar to control playback velocity. While the control line is being held in a particular position, the video information is played in the desired direction and at a velocity corresponding to the position of the control line along the slider bar. When the control line is released, it springs back to a central position of the slider bar, thereby causing playback of the video information to stop.

Another control technique is to use software radio buttons, which can be utilized to control the direction and velocity at which video information is played back in much the same manner that hardware buttons on tape players and VCRs control operating functions such as stop, step forward, forward, fast-forward, step reverse, reverse, and fast-reverse. In prior art systems, a radio button corresponding to each of these functions is displayed on the display with the video information, but such radio buttons are separate from any slider tools which also might be utilized. The function associated with a radio button is carried out when the radio button is selected by the mouse.

SUMMARY OF THE INVENTION

A preferred embodiment of the sequential information controller of the present invention comprises a video playback direction and velocity controller for use in combination with a computer and a video input device, such as a video cassette recorder (VCR), camera, or optical disc player. The controller regulates the play-

back of video information on the display of the computer through an interactive user interface, which works in combination with the computer's system software. The interface includes an interactive slider bar and an interactive control icon which are depicted within an operating window of the display. The slider bar includes both playback direction/velocity indicators and control buttons. The playback of video information is primarily controlled by manipulating the image of the control icon with respect to the image of the slider bar. The image of the control icon is that of a human-like hand, which holds the slider bar image in either its open or closed hand, depending on the desired playback mode, and can be moved along the length of the slider bar image by a control device, such as a mouse, to control the playback direction and velocity of the video information on the display. When the control icon's hand is open, the control icon is spring loaded from the stop playback position of the slider bar. To keep the control icon from automatically returning to the stop playback position when the control icon's hand is open, the user must continue to hold the control icon in position with the mouse. When the control icon's hand is closed, the control icon will remain at any position along the slider bar it is placed, until moved by operation of the mouse, or until returned to the spring-loaded mode and released by the mouse, thereby allowing it to spring back to the stop position.

The mode of the control icon is modulated by positioning the cursor over the control icon, selecting the control icon, and moving the cursor either up or down. Moving the cursor down, when the hand is open, closes the hand and freezes the playback velocity at the speed selected when the hand was closed. Moving the cursor up, when the hand is closed, opens the hand and spring loads the control icon. The control icon can also be rapidly moved to different positions by simply clicking on the slide bar with the mouse, regardless of the mode of operation.

The present invention offers numerous advantages over the prior art. First, the present invention is an improvement over stand-alone radio buttons because it allows the user to select playback velocity settings in between those offered by the radio buttons. Second, the present invention is an improvement over stand-alone, one-state sliders because the user doesn't need to guess to select standard velocity settings and because the user has the option of holding the control icon in position to maintain a desired playback velocity or setting the control icon in fixed position to select a desired playback velocity. Third, the present invention is an improvement over jog-shuttle potentiometers because it can be spring-loaded to return to the stop position when released and it offers numerous standard velocity settings which can be readily selected by the user. These and other advantages of the present invention will no doubt become apparent to those skilled in the art after having read the following detailed disclosure of a preferred embodiment of the present invention which is illustrated in the several figures of the drawing.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 is a block diagram representing a video editing system in accordance with the preferred embodiment of the present invention;

FIG. 2 is a representation of a video window and a control window as each might appear on the display of

a computer system utilizing the preferred embodiment of the present invention;

FIG. 3 is a representation of a control window illustrating the set mode of the present invention;

FIG. 4 illustrates methods for changing the position of the control icon with respect to the slider bar and for modulating the operating state of the controller in accordance with the preferred embodiment of the present invention; and

FIG. 5 illustrates a method for setting the playback direction/velocity of displayed video information according to the position of the control icon along the slider bar in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

In general, with reference to FIG. 1, a viewing or editing system for use in combination with the present invention is shown comprising a video source 10, a video processor/controller 12, and a computer system 14. The computer system 14 includes or is in communication with a mass storage device 16, a display device 18, and a control device 19, and could also include the video processor/controller 12 as an add-on board connected to its motherboard. Video information from the video source 10 is communicated to the video processor/controller 12 for conversion to digital signals, if necessary, and output to the computer system 14, where it is either displayed on the display 18 or stored in the memory of the mass storage device 16. Although the present invention is described as being used to control the display of video information, it could also be used in a similar manner to control the playback of any type of sequential visual or audio information, or even as a throttle control for an engine or other type of device which would require a linear controller to regulate the flow of fuel or other product (generically referred to as information from hereon) and would benefit from having fixed and tentative (springback) controller operating modes.

The type of computer system to be utilized in combination with the present invention is unlimited and includes any one of a broad range of different types of computers, although computer systems utilizing graphical user interfaces, such as the Macintosh® computer manufactured by Apple Computer, Inc., of Cupertino, Calif., are preferred. The computer system should typically include appropriate hardware and software for communicating with and/or controlling the source of the audio or visual information. The type of video source 10 to be utilized is likewise unlimited and can also be any of a large number of different types of devices, such as a video cassette recorder, optical disc player, certain types of video cameras, etc. It should also be noted that the controller of the present invention can be used to either control the actual operation of the video source (and therefore the playback of video information), through the computer system 14 and/or video processor/controller 12, or the playback of video information already stored in either the memory of the computer or the mass storage device 16.

With reference now to FIG. 2, the basic operation of the editing system and the controller will be illustrated. Upon initialization of the computer system 14 and the computer's system software, the user is presented with a number of operating windows on the computer system's display 18, such as the view window 20, video

window 22, and control window 24. In some computer systems, such as the Macintosh® computer, the user will also be presented with certain functional icons which allow the user to conveniently perform certain tasks within the computer without having to type in written commands. Each window of the computer also typically includes a close box 28 for closing the window when it is through being used, size boxes 30 for changing the size of the window, vertical scroll bars 32 for scrolling the images within the window up and down, and horizontal scroll bars 34 for scrolling the images within the window sideways. The techniques for creating functional icons and operating windows are well known in the art and need not be described here for an enabling disclosure of the present invention. See, Apple Computer, Inc., Inside Macintosh, Vols. I, II, and III, (1985), Addison-Wesley Publishing Company, Inc.

To display video information 35 from the video source 10 in the video window 22 of the general view window 20, the user must first indicate to the computer system 14 the type of video source to be utilized (i.e., VCR or laser-disc player) and the name and/or location of the desired video information to be displayed (i.e., the starting frame position or the title of the section of information to be edited). Once this task has been accomplished, the user can then control the display of the video information 35 through operation of the control device 19, such as a mouse, trackball, keyboard, touch screen, or any type of X-Y axis input device. The controller 36, which is operated by the control device 19, is shown generally within the control window 24. The controller 36 includes an interactive control icon 40 and an interactive slider bar 38 having a bar-shaped control dial, standard playback direction/velocity indicators, and a number of control buttons corresponding to the position of the playback direction/velocity indicators, including standard playback indicators and control buttons corresponding to fast-reverse 42, reverse play, 44, step-reverse 46, stop 48, step-forward 50, forward play 52, and fast-forward 54.

In the preferred embodiment of the present invention, the image of the control icon 40 is that of a human-like hand which holds the image of the slider bar in either its open or closed hand, depending on the desired playback mode of the controller (such as the tentative-set or fixed-set states further described below). The control icon 40 is moved along the length of the slider bar 38 by operation of the control device 19. The control icon 40 is typically manipulated by the well known technique of positioning the cursor 56 over the control icon (or similar type of graphic object depicted on the display), selecting the control icon either with the selection button of the control device 19 or a keyboard command equivalent, and dragging the control icon 40 along the slider bar 38 to a new position. It should be noted, however, that this method of manipulating the position of the control icon 40 is by no means exclusive, since the position of the control icon 40 could also be manipulated in a wide variety of manners depending on the type of computer system utilized and the types of peripheral control devices and software available.

Although the control icon 40 is depicted and described as a humanlike hand, it is important to note that the control icon 40 could have any of a wide variety of different designs or appearances. Preferably, the design of the control icon 40 should be such that it aids the user in determining the present operating mode or state of the controller, such as by depicting one shape when

operating in one state, and depicting a different shape when operating in a different state. If so desired, however, the icon could also be designed to have a single iconic image regardless of its operating state, but in such situations, some additional step should generally be taken to indicate the controller's state (i.e., changing the color of the slider bar when a state change occurs).

The open-hand control icon 40 depicted in FIG. 2 indicates that the controller is being operated in its tentative-set state. The controller is said to be operating in a tentative-set mode or state when the control icon's hand is open because the user can only temporarily set the playback direction and velocity of the video information (i.e., the control icon is spring loaded from the stop playback position 48 of the slider bar 38 and will automatically return to the stop playback position as soon as the user releases the control icon with the control device). To keep the control icon 40 from automatically returning to the stop playback position 48, the user must continue to hold the control icon 40 in position with the control device 19, or switch the operating mode of the controller to what is referred to as the fixed-set state. When the control icon's hand is closed, as illustrated in FIG. 3, the controller is said to be operating in a fixed-set mode or state because the position of the control icon 40 along the slider bar 38 remains fixed until the control icon is again moved by operation of the control device 19, or until the control icon is returned to its tentative-set state and released by the control device 19, thereby allowing it to spring back to the stop position 48. Hence, when the control icon 40 is operated in its closed-hand mode or fixed-set state, the control icon's position, and therefore the playback direction/velocity of the video information 35, remains fixed, until changed by further operation of the control device 19.

The position of the control icon 40, and therefore the playback direction/velocity of the video information, can also be rapidly changed, regardless of the controller's operating mode or state, by simply clicking (with the control device) on a different position on the slider bar 38. When a click is detected on the slider bar 38, the control icon is moved to the newly selected position and the playback direction and velocity are set accordingly. If the controller is operating in the tentative-set mode at the time of the click, the user will have to continue selecting the control icon (i.e., holding down on the selection button of the mouse), or switch the controller over to its fixed-set state, to keep the control icon at the selected position. If the controller is operating in the fixed-set mode at the time of the click, the control icon 40 will simply move to and stay at that selected position.

This particular feature works well in combination with the control buttons of the slider bar to quickly and accurately set a specific playback direction/velocity for the video information. Each control button of the slider bar 38 has an interactive display region, such as the region immediately surrounding one of the playback indicators, which is responsive to the position of the cursor 56. When the cursor 56 is moved to a position within one of these display regions and the selection button associated with the control device is activated, the controller is immediately commanded to perform a predetermined action, such as to move the control icon 40 to a position on the slider bar corresponding to the display region and to set an appropriate playback direction and velocity. For example, if a user wanted to playback the video information 35 at the standard, forward

play velocity, the user need only click on the forward play indicator 52 to move the control icon to that position, thereby causing the controller to move the control icon to that position and to set a playback direction and velocity corresponding to that indicator. The control buttons associated with the indicators 42 to 54 are most appropriately set up as radio buttons so that only one button out of the group of buttons can be active at one time. The areas of the slider bar between the interactive regions associated with the indicators function to indicate velocity gradations between the specific velocities associated with each control button along the slider bar 38 in each particular direction. Hence, if the user would like to set the forward playback velocity somewhere between the velocity associated with forward play indicator 52 and the velocity associated with fast-forward play indicator 54, the user need only click on the slider bar at, or move the control icon to, a position on the slider bar between the two indicators.

The operating state or mode of the controller, and hence the image of the control icon 40, can also be changed in a simple intuitive manner, which will be further described below. The controller's operating mode can be changed at anytime by selecting the control icon 40 and moving the cursor 56 either up or down by a sufficient amount. Moving the cursor 56 down, when the hand is open, closes the hand and freezes the playback velocity at the speed selected when the hand was closed. Moving the cursor 56 up, when the hand is closed, opens the hand and spring loads the control icon.

The operation of the controller can be further explained with reference now to FIGS. 4 and 5. The flow chart of FIG. 4 illustrates methods for changing the position of the control icon 40 with respect to the slider bar 38 and for modulating the operating state of the controller. The flow chart of FIG. 5 illustrates a method for setting the playback direction/velocity of displayed video information 35 according to the position of the control icon 40 along the slider bar 38.

As is illustrated in FIG. 4, upon initialization of the controller system, the controller is first prompted to look for a click (i.e., a mouse down command or selection command) somewhere on the controller image (meaning both the slider bar 38 and the control icon 40), block 70. When a click on the controller image is detected, the controller is first prompted, block 72, to determine if the cursor 56 is positioned over the control icon 40, or the slider bar 38. If the click occurred on the slider bar 38, then the controller is prompted, block 74, to move the control icon 40 to the position of the cursor 56 on the slider bar 38 and to set the playback direction/velocity accordingly. Once the control icon 40 has been set in block 74, or if the click occurred on the control icon, the controller is prompted, block 76, to follow changes in the X-coordinate position of the cursor 56 with the control icon 40 and to set the playback direction/velocity accordingly.

After setting the playback direction/velocity, the controller is then prompted, block 78, to determine whether any substantial change in the Y-coordinate position of the cursor 56 has occurred, thereby indicating a state change command. What is considered to be a substantial change is a matter of choice, but some latitude in the movement of the Y-coordinate position should be allowed so that the controller doesn't change modes too readily when the user is moving the control

