one entry, with each placement handle, and that entry indicates the handle of the workspace which includes the placement controlling the corresponding display object when the current workspace was entered. It is this placement which is subsequently updated in box 258 in FIG. 13, if necessary.

The controlling placement list created in FIG. 16 also plays a role in determining whether the placements in a given workspace are in the same order as when the current workspace was entered, in box 286 in FIG. 13. FIG. 17 shows a routine for testing whether the placement order has changed in a way necessitating the reordering of the placements in box 290.

The routine of FIG. 17 begins by creating a temporary list of the controlling placements which are in the workspace being examined, in box 400. These placements may be obtained from the controlling placement list created by the routine of FIG. 16, and should have the same order in the temporary list as they have in the controlling placement list. For each controlling placement on this temporary list, the routine retrieves the position of the corresponding window on the window list, in box 402. This can be done by finding the window list entry with a matching handle.

The routine then proceeds through the temporary list, with the test in box 404 determining whether any placements remain to be examined. If so, the next unexamined placement on the temporary list is taken in box 406. The test in box 408 then determines whether there are any uncompared placements after this placement. If so, the next uncompared subsequent placement is taken in box 410. The test in box 412 compares the window list positions of this uncompared placement with the placement being examined. If the position of the placement being examined is further down in the window list than that of the uncompared placement, they are not in the same order they were in when the current workspace was entered. It may not be necessary to reorder the placements, however, if the placements which are out of order do not overlap, as determined by the test in box 414. If reordering is not necessary, the test in box 408 is repeated for subsequent uncompared placements on the temporary list.

The routine continues through subsequent uncompared placements until all have been compared. Then, the routine returns to the test of box 404 to determine whether further placements remain to be examined. If not, the routine has gone through the entire list without finding it necessary to reorder, so that the result of the test in box 286 in FIG. 13 is that the placements are in the same order, as indicated in box 416.

If reordering is necessary, either because of the test in box 414 in FIG. 17 or because of the test in box 284 in FIG. 13, reordering begins with the top placement in the temporary list. If the routine comes directly from box 284, it is therefore necessary to create the temporary list as in boxes 400 and 402 before proceeding with this step. The test in box 432 determines whether placements remain on the temporary list. If so, the next remaining placement is taken in box 434, and the routine also finds the lowest list position of all the remaining placements, in box 436. If the test in box 438 determines that the placement taken in box 434 has the lowest list position, then this placement is correctly ordered. Otherwise, the placement with the lowest list position is exchanged with this placement in box 440, and the routine proceeds to the next remaining placement. When all the placements have been examined in this manner, the routine of FIG. 17 is complete.

When the routine of FIG. 13 has been completed, exit workspace procedure 204 performs any additional workspace exit procedures which are appropriate. Some of these procedures may be specific to the workspace being left, as in box 66 in FIG. 3. Each of these exit procedures is typically paired with an entry procedure in box 82 in FIG. 4, with the entry procedure setting up the system to operate in a particular manner while it is the current workspace and the exit procedure resetting the system to a neutral state in which it is ready to enter any workspace, the current workspace returning to its virtual state until it is next entered.

Some of the entry and exit procedures could, for example, determine how display objects appear within a workspace. An entry procedure could set up the workspace so that windows will be tiled in columns or will appear in fixed positions, with the paired exit procedure returning the window system to display windows in an overlapping mode. An entry procedure could change a connected directory, with the paired exit procedure restoring the previous connected directory. An entry procedure could retrieve a data structure specific to that workspace which supports particular operations in the workspace, with the paired exit procedure storing that data for subsequent retrieval during the next entry procedure in that workspace. An entry procedure could retrieve a variable to be used in the workspace, such as a handle for an input/output device such as a printer, with the paired exit procedure returning the system to a default variable.

Entry and exit procedures can also be used in implementing baggage features. For example, baggage operations can be invoked by a special sequence of signals from the user which also invoke the workspace exit procedure. Before proceeding with workspace exit, the baggage operations could provide a prompt asking the user to indicate the windows to be included in baggage, and the user could select each window, indicating whether it should be moved or copied into the next workspace entered. The workspace exit procedures could then store the data necessary to perform the requested move and copy operations upon entering the next workspace, and the workspace entry procedures could retrieve that data and delete and create placements accordingly.

At the completion of exit workspace procedure 204, the system is ready to enter another workspace or the overview. We turn now to enter workspace procedure 206.

3. Enter Workspace

Like exit workspace procedure 204, enter workspace procedure 206 may be invoked in a number of ways, each leading to substantially the same steps. One major objective of enter workspace procedure 206 is to produce a display closely resembling the display which existed when the user last exited from the workspace being entered. The same windows should be present, displayed in the same way and with the same apparent stacking order.

The implementation of exit workspace procedure 204 described above updates the workspaces data structure to include data necessary for regenerating substantially the same display at a later time. Enter workspace procedure 206, in regenerating the display, must reflect to an extent the steps taken in exit workspace procedure 204.

It does so in a manner which also follows the general sequence in FIG. 4.

FIGS. 18A and 18B illustrate a procedure for entering a workspace which reflects the steps taken in the procedure of FIG. 13. The basic strategy of FIGS. 18A and 18B is to walk through the workspace data structures, displaying the window corresponding to each controlling placement from the bottom to the top.

The enter workspace procedure of FIG. 18A begins in box 450 when the exit workspace procedure is completed in response to a command to enter another workspace or when the overview is exited to enter a workspace. In either case, the identifier of a workspace to be entered will be provided, and certain other data will have been stored in exiting the previous workspace. The step in box 452 retrieves some of that data and takes preparatory steps based on it, such as determining whether the workspace being entered is different than the previous workspace so that a back door should be created, in which case a display system object for a back door is set up and linked by a placement to the workspace being entered. Other workspace entry procedures in box 452 may include creating placements for baggage from the previous workspace and setting up the system to operate in the workspace being entered, as discussed above. A list of workspaces visited during the recursive walk, designated the Seen list, is set empty in box 454. The step in box 456 then calls the recursive procedure EnterProc, providing the current workspace's handle and the Seen list. Upon receiving the Seen list back, the step in box 457 calls EnterProc, providing the handle of the pockets workspace and the Seen list. Upon completion, the workspace has been entered and display system operations resume in box 458. The workspace system in effect becomes dormant until the next sequence of user signals invoking workspace operations.

EnterProc procedure in FIG. 18B begins with a call with the handle of a workspace, WS, and the Seen list. The test in box 561 determines whether WS is in the Seen list. If not, WS is added to the Seen list in box 462 and the corresponding data structure is accessed in box 464. The background specification from that data structure is accessed in box 466 and used to generate any specified additional background features.

The test in box 468 determines whether any unexamined inclusions remain in the current workspace. If so, the next unexamined inclusion is accessed in box 470, and EnterProc is recursively called with the workspace handle from that inclusion, in box 471. Upon completion, the test in box 468 is repeated.

When no more inclusions remain in the current workspace, the test in box 472 determines whether any unexamined placements remain. If so, the next unexamined placement is accessed in box 473, and its handle is used in box 474 to access the corresponding display system object. The data from the placement is loaded into that display system object in box 476, and the display system object proceeds to display the appropriate display object with the display characteristics indicated in the placement. This effectively moves the display object back from the hiding place onto the display or otherwise makes it visible and changes its display characteristics in accordance with the placement. This step should be done without interrupting the operations of the display system object, so that continuity is maintained.

When all the placements in the current workspace have been examined, the procedure returns the Seen list, in box 478. If the test in box 461 determined that WS had already been visited, the procedure would immediately return.

As noted above, enter workspace procedure 206 can occur not only after exit workspace procedure 204 but also after exit overview procedure 214. Similarly, exit workspace procedure 204 can lead not only to enter workspace procedure 206 but also to enter overview procedure 210. We turn now to consider the overview-related procedures and other features of the overview in more detail.

E. The Overview

Some features of the overview are discussed above in relation to FIGS. 5–7. FIG. 11 shows the sequence of commands leading to the overview, and FIG. 14 shows a sequence of user signals which result in leaving a workspace and entering the overview.

The overview could be implemented in various ways. For example, the overview can be implemented as a unique workspace which has a unique set of display objects and, unlike the other workspaces, does not appear in the overview. It could also be implemented as a workspace like the other workspaces, so that the display objects in it could be moved and copied to other workspaces. The following description generally is based on the implementation of the overview as a unique workspace whose display objects are based on data stored in an overview cache which contains current versions of the workspace and window pictograms, to save time in entering the overview.

After briefly discussing further details of workspace and window pictograms, we will cover the operations available in overview interface mode 212 in one implementation of the overview according to the invention.

1. Pictograms

The layout and detailing of pictograms is discussed briefly in relation to boxes 124 and 126 in FIG. 6. FIG. 19 illustrates in more detail the layout and detailing steps of FIG. 6.

The enter overview procedure of FIG. 19 begins in box 480, and initial steps as shown in boxes 120 and 122 in FIG. 6 include providing the overview background and title as well as the overview tools. The procedure then accesses the overall data structure which includes all the workspace data structures, in box 482. The workspaces are listed and counted in box 484. This list, unlike the workspace list discussed in relation to FIGS. 15A and 15B, may be based on alphabetical order, and may be created simply by sorting the overall data structure so that the workspace data structures are ordered alphabetically by workspace name. Then, in step 486, the sizes and positions of the workspace pictograms are determined based on the number of workspaces. In the example of FIG. 5, with eleven workspaces, positions and sizes are assigned so that three rows of four workspace pictograms each will fit within the available display area in the appropriate order, so that an area for the twelfth workspace pictogram is open. This corresponds to the layout step in box 124 in FIG. 6.

The details of each workspace's pictogram may be stored in an overview cache, as mentioned above. Therefore, to obtain the pictogram, the procedure of FIG. 19 checks whether the pictogram in the overview cache is accurate and, if not, recreates the entire pictogram in a manner very similar to the procedure of FIGS. 18A and 18B. The test in box 488 determines whether any unexamined workspaces remain on the list of workspaces, and if so the next workspace's data

structure is accessed in box **490**. The test in box **492** determines whether the workspace has been changed in a way which will affect the pictogram, testing whether the workspace was marked changed during an execution of the procedure of FIG. **13**. If so, the procedure retrieves the cached pictogram and displays it at the appropriate position and size within the overview in box **494**.

If the test in box **492** determines that the workspace has been changed, the procedure first sets up a blank pictogram with a title, in box **496**. The step in box **500** lists the controlling placements as in FIGS. **15A**, **15B** and **16**. In the process, however, the background of the blank pictogram is painted, as in box **466** in FIG. **18B**.

If the test in box **510** determines that unexamined placements remain on the controlling placement list, the next unexamined placement is accessed in box **512**. It is desirable to include in some of the window pictograms details about the display system objects which provide the corresponding windows, and this requires that the display system objects be accessed. As discussed in more detail below, an access of this nature is outside the normal display system operations, and requires special functions which are tailored to the type of display system object being accessed. A window registration system, discussed in detail below, includes a set of functions for each registered type of display system object, and one function for each type is capable of recognizing whether a given display system object is of that type. In box **514**, the Recognize function for each of the types is called, together with the handle from the placement being examined. If any of the Recognize functions recognizes the display system object, in box **516**, another function for that type, the Present function, is called in box **518**. This Present function is tailored to retrieve from a display system object of that type any information to be displayed in its pictogram. If the display system object is not recognized, however, a blank pictogram is provided, in box **520**.

The step in box **522** then displays the pictogram obtained in accordance with the position and size data in the placement. If the position of the placement is outside the boundaries of the workspace pictogram created in box **496**, this step may also involve reshaping the workspace pictogram to include a virtual part.

When all the pictograms in the workspace pictogram have been displayed, the procedure returns to the test in box **488**. And when all the workspaces on the alphabetical list have been visited, the overview is complete, and the overview interface mode **212** is entered in box **530**. We turn now to the details of that mode.

2. Overview Interface Mode

When the overview is presented to the user, various operations are available which would be impossible or at least difficult when one of the workspaces is displayed. As noted above, the overview allows the user to navigate from one workspace to any other workspace, and assists the user in selecting a destination by presenting information about each workspace. Furthermore, if workspaces are implemented with a preexisting display system, the overview provides an opportunity to perform a number of operations not available during display system operations, including manipulations of placements, of workspaces and of collections of workspaces.

In one implementation, all of the pictograms are windows of the window system, with the window pictograms being superimposed on the workspace pictograms, and with all of the pictograms being displayed when the overview workspace is displayed. Each of these windows is provided by a corresponding display system object which receives and responds to user signals relating to that window. In this case, the overview interface mode is simply a special case of normal display system operations, and user signals relating to the pictograms and other overview display objects invoke appropriate procedures. The overview could be implemented in a number of other ways, however, and need not be part of normal display system operations.

FIG. **5**, discussed above, shows overview **100**, which includes some additional features. As mentioned immediately above, each workspace pictogram has a background, the background of control panel **102** being closely spaced parallel diagonal lines, for example. Window pictogram **530** and other window pictograms have additional detail based on data retrieved from the underlying display system object by the present function. Horizontally extended workspace pictogram **532** includes, in addition to that part of a workspace which is presented on the display, an additional area which shows a part of window pictogram **534** which extends beyond the displayed part, occupying a virtual part of that workspace. Similarly, vertically extended workspace pictogram **536** includes an additional area which shows a part of window pictogram **538** in a virtual part of the workspace.

Keyboard command buttons **540** provide a variety of overview interface operations. In general, overview **100** is designed to give the user an indication of each available operation and of each change of state which occurs while in overview interface mode. The actual implementation of overview **100** affects the manner in which a user invokes an operation as well as the display features indicating execution of that operation. For example, in one implementation a user selection of one of keyboard command buttons **540** does not directly invoke the corresponding operation, but rather results in a reminder in prompt window **114** of the keyboard sequence to invoke that operation. And whenever that operation is invoked, the corresponding keyboard command button is inverted, as shown for "Edit" button **542**, and remains dark until the operation is completed.

We turn now to consider the operations which correspond to keyboard command buttons **540**.

Some keyboard command buttons correspond to navigational operations. The "Enter" button, for example, corresponds to the basic navigational operation of entering a workspace. Therefore, when the enter operation is invoked and a workspace is selected, an exit overview procedure like that described in relation to FIG. **7** is performed and then an enter workspace procedure as in FIG. **18**.

In order to select a workspace, the user moves a pointer using a mouse or other pointer control device until it points to the workspace pictogram corresponding to the workspace to be selected. Since each workspace pictogram is likely to contain one or more window pictograms, it is necessary to distinguish somehow between a workspace selection and a window selection when the pointer is pointing to a window pictogram. If the mouse has more than one button, one button can be used to indicate selection of the window pictogram and another to indicate selection of the workspace pictogram in which the window pictogram is located. Upon receiving a window pictogram selection, the corresponding placement is accessed based on information in

the overview cache which indicates which placement corresponds to each region within each workspace pictogram.

A number of keyboard command buttons **540** correspond to operations which manipulate placements, and a placement to be manipulated is selected by selecting the corresponding window pictogram in the workspace pictogram of the workspace which contains it. The "Move" button corresponds to the operation of moving a placement to a selected location within the same workspace by modifying the placement or from one workspace to another selected workspace by deleting one placement and creating another linked to the other workspace. The "Copy" button with selection of a window pictogram corresponds to the operation of creating another placement linking the same display system object to another selected workspace. The "Shape" button corresponds to the operation of modifying a placement to a selected size, the size being selected by adjusting an outline of the window using the mouse. The "Delete" button with selection of a window pictogram corresponds to the operation of deleting the corresponding placement.

An operation which manipulates a placement, but which does so to assist navigation, is the expand operation, corresponding to the "Expand" button. The expand operation displays the full size current contents of a selected window at a location on the workspace pictogram to which it is linked by the selected placement. If the expand operation is invoked with selection of a window linked to an included workspace, the current contents are thus displayed full size at a position based on the placement but on the included workspace's pictogram rather than the workspace pictogram in which the selection was made. Since all the windows will be at the hiding place during overview interface mode, the expand operation can be performed by moving the selected window from the hiding place to the workspace pictogram to which it is linked. The expand operation is especially useful in finding a window based on its current contents.

Another placement manipulation operation which may be useful in navigation corresponds to the "Shares" button. This operation, after determining the selected placement, goes through the overview cache and finds all the other placements which contain the same handle. The window pictogram corresponding to each such placement is then made more visible, by blinking or the like, so that the user can identify all the workspaces that share that window.

Some of the workspace manipulation operations correspond to placement manipulation operations. For example, a copy or delete operation can be performed on a selected workspace by copying or deleting that workspace's data structure. In order to copy a workspace, however, it is necessary to assign a different name to the copy, so that the user will be asked to provide a name during the copy operation. The overview will be laid out again to accommodate the copied workspace. The "New" button corresponds to a similar operation which creates a new workspace and asks the user to provide a name. A variable can be set up to provide a default background specification and a list of inclusions that will automatically be included in a new workspace created in this manner. Similarly, the "Rename" button corresponds to an operation which asks the user to provide a new name for an existing workspace.

The "Edit" button corresponds to a workspace manipulation operation which permits the user to edit a workspace's data structure. It does so by displaying a description of the data structure within the corresponding workspace pictogram, as shown in pictogram **550** in FIG. **5**. Pictogram **550** is thus a window in which interactive editing can be done, with the available commands appearing in adjacent edit menu **552**. When editing is completed by an appropriate user signal, the workspace pictogram is generated again in accordance with the changed data structure. More than one workspace could be edited in this manner at a time, so that data could be copied from the description of one workspace's data structure to another's.

The remaining keyboard command buttons correspond to workspace manipulation operations which can assist in navigation and in understanding the connections between workspaces, either through doors or through inclusions. Each connection may be shown by a link which is a line with one end a light circle and the other end a dark circle. The "Doors" button corresponds to an operation which shows where each door in a selected workspace leads, the light circle of each link being at the door and the dark circle being on the workspace pictogram of the workspace to which that door leads. The "Doors to" button corresponds to the converse operation, linking all doors in other workspaces which lead to a selected workspace. The "All doors" button corresponds to an operation which links all doors and their destinations. The "Includes" button corresponds to an operation which links a selected workspace to all workspaces it includes, the light circle being in the including workspace and the dark circle in the included workspace. The "Included in" button corresponds to the converse operation, linking a selected workspace with all the workspaces which include it. And the "All inclusions" button corresponds to the operation of showing all inclusion relationships in this manner. In order to reduce viewer confusion, the links generated by the all doors and all inclusions operations are curved so that they do not overlap.

In addition to keyboard command buttons **540**, save button **108**, restore button **110** and augment button **112** invoke overview interface operations. The corresponding operations all relate to manipulation of a selected collection of workspaces. Because these operations relate to the more general topic of groups or suites of workspaces, we will treat them in the following section.

F. Suites of Workspaces

Generating a useful group or suite of workspaces involves a sufficient amount of effort that it is often worthwhile to preserve them. This requires not only that the windows or other display objects in the workspaces be preserved, but also that the placements, inclusions and other data in the workspaces' data structures be preserved so that the effort of recreating those workspaces is not necessary. For example, a user may wish to backup on a file server a suite of workspaces which has been created in a work session. Or a user who has created a useful suite of workspaces may store them on a file server for copying by other users. This may be especially useful for transferring a useful suite of workspaces from an expert to each of a group of novices. Once a suite of workspaces has been stored as a file, it can be mailed or copied like any other file.

Useful operations on suites of workspaces include the save, restore and augment operations invoked respec-

tively by save button **108**, restore button **110** and augment button **112**. Although these buttons are presented in the overview and are thus a part of the overview interface mode, some of the operations they invoke when selected could be available at other times through appropriate user signals. The save operation, for example, stores a selected group of workspaces in a designated file, together with the necessary data to regenerate those workspaces and the display objects they contain. Therefore, the save operation is useful at any time that it is desirable to back up a suite of workspaces or to make a suite of workspaces available for mailing or copying. The restore and augment operations, on the other hand, take a suite of workspaces from such a file and regenerate and present the workspaces to a user, the restore operation replacing the user's preexisting workspaces and the augment operation adding the regenerated suite of workspaces to the preexisting workspaces. These operations may be thought of as file operations because each involves storing data in a file or retrieving it from a file.

The data stored in or retrieved from a file by a file operation differs in one significant respect from the data in a workspace data structure: It includes not only the display characteristics from the underlying display system objects, but may also include data about the applications and other procedures which a display system object calls and the data structure a display system operates on in providing the contents of a window or other display object. This additional data may be necessary in order to recreate the display system object so that it can provide its display object within a recreated workspace. But the specific additional data which should be included in the file for a given display system object depends on the characteristics of the object. For some objects, no data should be stored, because the workspace system of any user who augments from the file will already contain an equivalent object, such as the user's mail facilities. In general, the context-dependent characteristics of the objects, including details like the user's name, should not be stored. For other objects, the entire display system object should be stored, because it is necessary for a task which the user will perform and the user's workspace system is unlikely to have an equivalent object. Indeed, a primary purpose of storing a suite of workspaces may be to provide a set of display system objects to an inexperienced user.

After discussing a window registration system which assists in obtaining the appropriate data from display system objects, we will discuss save, augment and restore operations which employ the window registration system.

1. Window Registration System

In order to obtain the appropriate data from each type of display system object, it is helpful to categorize the objects into types and provide a set of procedures for each type. For that purpose, one aspect of the present invention is to provide a window registration system, mentioned above in relation to providing window pictogram details which are similarly based on data from each type of display system object. This window registration system facilitates the process of retrieving data from display system objects by providing a set of functions appropriate for each of a number of types of display system object.

A window registration system is not necessary for the ordinary workspace operations described above because those operations involve data in the display system objects which are uniformly retrievable. In other words, the display system objects are structured so that the position, size, visibility (which may be collapsed into position), drop shadow data, shrinking data and other miscellaneous display characteristics are all retrievable by a procedure which is independent of which display system object is accessed. For example, this data could all be collected at the beginning of a data structure following the object's handle, so that it can be accessed and retrieved with the handle and without knowing the subsequent structure of the object. In fact, this is how conventional window systems are typically designed. But the window registration system is necessary to access and retrieve data from the subsequent part of the object because that part typically has a structure unique to that type of object.

FIG. **20** shows the relation between software components within a user interface system **570**, including window system **572**, user invoked file procedures **574** and window registration system **576**. Underlying window system **572** provides the actual user interface, receiving user signals and presenting display features to the user. When the signals from the user invoke one of the file procedures, such as save, restore or augment, window system **572** calls user invoked file procedures **574**, indicating which procedure is requested and providing any data necessary to that procedure's execution. During its execution, that procedure may require data from one of the display system objects in window system **572**. That data is retrieved by calling appropriate functions from window registration system **576**.

Window registration system **576** may be implemented in a number of ways. One implementation is a window type table containing a window type entry for each type of window. Each type entry includes a type handle or name which can be used to access that entry directly and a number of function handles which can be used to access the functions for that type. The display system object for each window of a given type responds to the same set of functions, so that the entry for that type can be set up by providing the type name and function handles for those functions.

The functions in the type table for each type may include, for example, a Recognize function, an Abstract function, a Recreate function and a Present function.

As discussed above in relation to box **516** in FIG. **19**, the Recognize function of a given type, when it receives the handle of a window, accesses the corresponding display system object in window system **572** to determine whether it is of that type. The Recognize function may do this by interrogating the object for a particular function or other property, such as whether the object allows application code to be run in a window. If the Recognize function recognizes the object, it returns the handle of its type to the procedure which called it, so that the procedure can then call other functions for that type. If the Recognize function does not recognize the object, it returns a response indicating non-recognition, such as a Boolean value of false.

The Abstract function of a given type provides an abstract description of a display system object which can be stored in a file. The Abstract function receives the handle of a display system object, accesses that object to obtain the data needed to recreate that object subsequently, and provides the data in an appropriate form, including the type name. The Abstract function could, for example, retrieve characteristics of the data presented within the window provided by that object or

other results of the current execution of an application called by that object. As noted above, the Abstract function should not retrieve context-dependent data, because that data will be different when the display system object is set up in a different environment.

The Recreate function of a given type reconstitutes a display system object of that type from its handle and the abstract description provided by the Abstract function. That abstract description may create the object from scratch or it may call applications and other procedures which are available in the user's environment to assist in recreating the object. It may not create a new object, but may rename an existing object or may obtain the handle of an existing object and provide that handle as the object's handle. The result is an object which fulfills a role in the workspace system.

The Present function of a given type, as mentioned in relation to box **518** in FIG. **19**, provides data used in presenting a window's pictogram in the overview. As shown in FIG. **5**, a number of the window pictograms are labeled with words like "TEdit", "Exec" and "Snap". These words refer to applications called by the display system objects which provide those windows, and the Present function of the appropriate type accessed each such display system object to find the name of the application called. This is an example of the type of data the Present function may retrieve.

In general, window registration system **576** can handle as many functions as appropriate. The functions described above are illustrative of the kinds of functions which are useful. The type table provides rapid selection of the appropriate function.

The type table is created during initialization in box **220** in FIG. **12** and is, in effect, supported by an application in the display system that contains all the functions of all the recognized window types. This application, referred to as the window type registrar, also includes loading code which sets up the type table to include the type name and function handles for each of the recognized types. Furthermore, if a new type of display system object is added to the window system, its functions must also be written together with loading code for calling the window type registrar to add a new entry to the type table containing the new type's name and the handles of its functions.

The above-described window registration system is used primarily in relation to the save, restore and augment operations.

2. File Operations

Three file operations are provided in overview interface mode **212** by save button **108**, restore button **110** and augment button **112**. FIG. **21** illustrates the save operation, while FIG. **22** illustrates the restore and augment operations.

The procedure in FIG. **21**, which implements the save operation, begins in box **600** when a save command is received. The procedure prompts the user to select workspaces to be included in the saved file, in box **602**. The test in box **604** determines whether the user selected any workspaces. If not, all the currently existing workspaces are selected in box **606**. In either case, a list of the selected workspaces is generated in box **608** for subsequent operations. In box **610**, the procedure prompts the user to provide a file name, and a file with that name is then set up in box **612**.

At this point the procedure of FIG. **21** begins to go through the list of workspaces generated in box **608**. The test in box **614** determines whether any workspaces remain on the list. If so, the next workspace's data structure is accessed in box **616**. The name or other handle of the workspace is written to the file being created locally in such a way that subsequent data item can be written behind it to complete a data structure depending from it. Then begins the process of writing that workspace's data structure to the file.

The test in box **620** determines whether that workspace's data structure has any remaining unexamined placements. If so, the next unexamined placement is accessed in box **622** and its handle is compared with the handles of placements on a list of placements whose corresponding display system objects have already been abstracted and written to the file, in box **624**. If the handle is not yet on the handle list, it is added to the list in box **626**.

The procedure then calls the Recognize function of each of the object types in the window registration system in turn. If any of the types recognizes the object, as determined in box **630**, the Abstract function of that type is called in box **632** to obtain an abstracted description of the object from which it can be recreated. That description is then set to be the descriptor of the object, in box **634**. If the object had not been recognized, the descriptor would be a code indicating that the object is of an unrecognized type and other appropriate data such as its title and other data from which it could be recreated if its type were registered, in box **636**. In either case, a file handle to be used to access the descriptor in the file is generated and added to the handle list in a pair with the handle from the placement, in box **638**. Then, the file handle is written to the file as a placement in the data structure of the workspace being examined, and a pair including the file handle and the descriptor is also written independently to the file, in box **640**. Then the procedure returns to the test of box **620**.

If the test in box **624** determines that the handle from the placement is already on the handle list, the corresponding file handle is retrieved from the handle list That file handle is then written to the file as a placement in the data structure of the workspace being examined in box **642** before returning to the test of box **620**.

When all of the placements in a workspace have been examined in this manner, the test in box **644** determines whether the workspace has any unexamined inclusions. If so, the next inclusion is accessed in box **646** and its handle is written to the file as an inclusion in the data structure of the workspace being examined in box **648** before returning to the test of box **644**. When all the inclusions have been written in the manner, the procedure returns to the test of box **614**.

When the test in box **614** determines that all the workspaces on the list have been examined, the file is complete. The completed file is then stored, in box **650**. The procedure of FIG. **21** could also be structured to create an entire list of handles with descriptors without writing any data to the file. In that case, the file would be first constructed to include the handle/descriptor pairs, after which the procedure would scan through the workspace list to write out each workspace's name, followed by its placements and inclusions. In either case, when the file is stored in a file server or other appropriate location, it can be copied, mailed or otherwise accessed for subsequent operations.

As can be seen from the procedure of FIG. **21**, none of the references to display system objects is permitted to dangle. In other words, every handle which is found in any of the placements in the selected workspaces

results in the writing of a file handle/descriptor pair to the file. But any inclusion with a handle of a workspace which is not selected is permitted to dangle, meaning that the procedure makes no effort to add the included workspace to those selected or to change its handle to the handle of a selected workspace.

FIG. 22 illustrates procedures implementing the restore and augment operations which basically reverse the process in FIG. 21. The procedures of FIG. 22 begin in box 640 when a restore or augment command is received. The procedure prompts the user for the name of the file to be restored or augmented in box 642, and that file is retrieved in box 644. The file handle/descriptor pairs stored in it are used to recreate the corresponding display system objects by calling the appropriate Recreate functions in box 646. The descriptor includes, in each case, the type of the object, and that type's Recreate function can be called to recreate the object. It may do so by setting up any necessary applications and by taking other steps. But if the type is unrecognized, a default Recreate function is called to create an object of unregistered type. That object retains any other data included in the descriptor, and if that object's type is later added to the window registration system that object can be recreated.

As each object is recreated in box 646, it receives a display system handle, and that handle is used to create a list of pairs, each including a display system handle and a file handle, in box 648. This list is then used to change all the file handles in the workspace data structures to display system handles, in box 650. This step can be performed as the workspace data structures are retrieved from the file, so that when it is completed the workspace data structures have all been retrieved. At this point the inclusions may have dangling references, as noted above, if there were dangling references in the file.

The next step which is taken depends on whether the command was to restore or to augment, as shown by the branch in box 652. If the command was to restore, the preexisting workspaces are deleted in box 654 and the overall data structure from the file becomes the overall workspaces data structure. The procedure exits and reenters the overview in box 656 so that the overview display can be recreated and stored in the overview cache. Then the procedure reenters the overview interface mode in box 658.

If the command was to augment, the test in box 660 determines whether any of the workspace data structures from the file remain to be examined. If so, the next workspace data structure is accessed in box 662. Its name is compared with the names of preexisting workspaces in box 664, and if it is a duplicate, a new name is assigned to the workspace data structure from the file, in box 666. Finally, the workspace data structure is added to the preexisting overall workspaces data structure in box 668. When all the workspaces from the file have been added to the preexisting overall workspaces data structure, the procedure exits and reenters the overview in box 670 so that the overview display can be recreated and stored in the overview cache. Then the procedure reenters the overview interface mode in box 672.

The usefulness of the operations described in relation to FIGS. 21 and 22 can be better understood by considering the specific problems of how to set up workspaces for an inexperienced or naive user; how to enable a user to configure a suite of workspaces; and how to deliver applications within workspaces. An experienced or expert user creates a set of workspaces which includes a number of workspaces with windows appropriate for delivery. The application behind each window need not contain extensive data, since the function of the workspaces is to provide a blueprint or empty suite of workspaces including applications which a subsequent user will fill in with appropriate data. The expert then invokes a save operation on those workspaces, storing them as a suite of workspaces in a file on a file server. The recipient user, whether an inexperienced user or any other user receiving the saved suite of workspaces, then accesses the file server and invokes an augment operation on the file, causing the suite of workspaces to be added to the preexisting workspaces.

We turn now to a useful technique for delivering a suite of workspaces.

3. Catalogue

Delivery of a suite of workspaces may be implemented using a catalogue metaphor, as illustrated in FIGS. 23 and 24. In order to make the catalogue available to the user, a catalogue workspace is provided which includes a display object like catalogue cover 680 shown in FIG. 23. This catalogue workspace could, for example, be the initial workspace created during system setup as described in relation to FIG. 11, so that catalogue cover 680 would be presented to the user at the beginning of display system operation. Catalogue cover 680 bears the term "ROOMS" which, as noted above, is a trademark of Xerox Corporation used in connection with an implementation of the invention.

Catalogue cover 680 may be implemented as a window which includes a basic piece of bit map, together with a number of buttons attached to the basic bit map. As used here, the term "button" means a piece of bit map which can be positioned at the edge of or at a point within a larger piece of bit map and which, when selected by the user, results in the execution of an associated piece of software. The buttons attached to catalogue cover 680 are shown as tabs at the right side of FIG. 23, including introduction tab 682, suites tab 684, ROOMS tab 686 and decor tab 688. Each of these tabs, when selected, results in the execution of a corresponding routine.

FIG. 24 illustrates the result of selecting suites tab 684. Catalogue cover 680 is removed from the display and replaced by suites pages 690, another basic piece of bit map. Tabs 692, 694, 696 and 698 are buttons attached to suites pages 690 and correspond in function to tabs 682–288 in FIG. 23. Box 702 is also a button positioned within suites pages 690, and selecting it results in an augment operation which adds a suite of workspaces called Office—1 to the preexisting workspaces. At the conclusion of such an operation, the windows within the selected suite of workspaces will be opened and the workspaces will all be accessible. Suites pages 690 includes a piece of bit map 704 from one of the workspaces in Office—1, to suggest the tasks for which that suite of workspaces is useful. Similarly, Box 706 is a button within suites pages 690, and selecting it results in an augment operation adding a suite of workspaces called Techdesk, represented by a piece of bit map 708 from one of its workspaces.

Selecting introduction tab 682 or 692 results in introduction pages which explain to the user how to make use of the options presented within the catalogue metaphor. Selecting ROOMS tab 686 or 696 results in workspaces pages within which buttons are presented which

when selected, augment a single workspace rather than a suite of workspaces. Selecting decor tab 688 or 698 results in decor pages within which buttons are presented which the user can select to modify the background pattern of a selected workspace. To further follow the catalogue metaphor, additional buttons with a piece orbit map which looks like a page corner can be used to permit page turning within each section of the catalogue.

The catalogue metaphor proves particularly useful as a technique for delivering applications in a system running workspaces, because of the augment operation. But it could be applied in other systems as well. One general feature of the catalogue metaphor which makes it useful is the uniform access it provides to all variations possible within the retrieved applications, including backgrounds and other display characteristics of the workspaces and the various suites of workspaces which are available.

In order to reach the catalogue, it may be appropriate for the user to go through preliminary steps, since not all users will want the catalogue. When the workspace system enters the initial workspace, a menu may be available to set up workspaces directly or to create doors to enter the catalogue or other special workspaces which assist a new user in understanding multiple workspaces.

We turn now to consider other characteristics of different implementations of the invention.

G. Implementations of Multiple Virtual Workspaces

The invention has been implemented in two distinct systems. One is called "Desk Tops" and the other, as noted above, is called "ROOMS".

Desk Tops is implemented in the Cedar environment developed at the Palo Alto Research Center of Xerox Corporation. Desk Tops creates several virtual desktops, each of which appears to be a normal desktop when it is displayed. Desk Tops makes use of viewers, which are containers which resemble windows. A desktop is a special viewer which can contain a configuration of viewers, the viewers in a desktop being typically tiled in two columns on the screen.

Desk Tops provided for multiple desktops and commands for moving back and forth between desktops. Facilities are also available for moving and copying viewers between desktops and for placing viewers in iconic form. A particular viewer can be in more than one desktop, and it can have a different size and position in each desktop. The data structure for each desktop contains a placement in the form of a pointer to each contained viewer and the size and position of that viewer in that desktop. When the user switches between desktops, the current configuration of viewers is saved in the current desktop before moving the viewers in the new desktop onto the screen.

Desk Tops is implemented so that a number of functions relating to a given type of viewer are called using a registration system resembling the window registration system described above. In addition, a viewer is made invisible by resetting a boolean value in the display system object which provides it. Also, whenever a program prints to a viewer that is not on the screen, Desk Tops moves that viewer to the current desktop before printing to it, so that the user will be able to see what is printed.

Desk Tops does not have an overview, but if a user deletes a desktop which contains a viewer which is not on any other desktop, that viewer is moved onto the current desktop in iconic form so that it will not be stranded.

Deleting a viewer may result in deletion of that viewer in all the desktops containing it, but not in deletion of the placements.

ROOMS is implemented in the version of Lisp marketed under the trademark Interlisp-D by Xerox Corporation. Interlisp-D has a number of unusual characteristics which affect the manner in which ROOMS is implemented. For example, all of the display objects in Interlisp-D are windows, with an icon being a special type of shrunken window; the underlying display system object remains active even in iconic form. The various windows in Interlisp-D can overlap freely, but a user selection in a given window will bring that window to the top of the display stack, through a default procedure.

Another characteristic of Interlisp-D windows is that a group of them can be attached so that they will be treated as a single display object. In that case, one of the windows controls the entire group, so that the controlling window becomes the window whose handle is used in the placement for that group. When the group is shrunk, it creates a single shrunken icon, which is a new window, so that the group of windows becomes invisible. Even then, however, the controlling window remains the window of interest. When performing workspace operations involving such a group, it is necessary to include in the placement data about whether shrunk, the size of the shrunken form, drop shadow and so forth, but it is not necessary to include the list of attached windows in the placement.

In Interlisp-D, it is possible to lock a window to a particular location on the screen. On the other hand, the only practical way to make a window invisible is to cause its display system object, also called a window, to provide its output to an invisible location, like the hiding place described above. In order to move a locked window to the hiding place, it is necessary to override the lock until the window is moved and then relock it.

ROOMS includes the overview, catalogue, sharing of display system objects, inclusions of workspaces and other features described in the preceding sections.

These two implementations each apply the invention to a preexisting display system. An implementation of the invention with such a display system may require minor modifications to fit the particular features of the display system. But the invention should be applicable to the conventional display systems now available.

H. Miscellaneous

Many other modifications, variations and extensions of the invention will be apparent to those skilled in the art from the above description. In many areas, a number of options are available for implementing a given feature, and the choice of an implementation depends on specific design factors.

As described above, a number of different sequences of user signals can lead to a workspace exit procedure. The background menu can be particularly important when in a workspace because it always permits the user to enter another workspace or the overview, so that the user can never be trapped in a workspace and unable to leave. In the overview, however, the background menu could be suppressed throughout the overview or within each workspace pictogram, because other steps can always be taken to leave the overview, such as using the enter operation to enter a workspace.

Another technique which protects the user from getting trapped is the back door. A back door can be automatically created whenever the user exits one workspace and eaten another, either directly or through the overview. No back door would ordinarily be created, however, if the user reenters a workspace from itself. The workspace exit and entry procedures will determine whether to create a back door and, if so, where to locate it, according to any appropriate protocol.

Like other doors, the back door is a display object which, when selected, results in a switch out of the currently displayed workspace and into another workspace. A door is usually labelled with the name of the workspace to which it leads, and a display object analogous to a door could also be provided for entering the overview.

The background specifications in the workspace data structures can include various commands resulting in a distinctive appearance for each workspace. The whole background or a region can be shaded or tesellated. A frame can be created around a region with a specified width. A piece of bitmap or a string of text can be located within the background. As described above, background commands can be made conditional, as on whether the workspace is included in another workspace in the current display.

The overview also presents rich possibilities for variation. The operations available in the overview interface mode may be extended to include any convenient operation which assists the user in the use of the system, whether related to navigation or to another purpose.

An important potential extension of the invention is the sharing of a workspace by two or more users. This sharing could be done in a number of ways. Serial sharing could be provided by storing the workspace on a shared file server with a lock which the user would set when using the workspace and unlock when leaving the workspace. The user could use the save operation in the process of leaving the workspace to preserve the current state of the display objects. Simultaneous sharing could also be provided with a shared file server or with a remote procedure call. Simultaneous sharing of a workspace could be implemented by sharing all the windows in a workspace, along the lines of a multi-user interface, or by sharing the workspace itself.

Although the invention has been described in relation to various implementations, together with modifications, variations and extensions thereof, other implementations, modifications, variations and extensions are within the scope of the invention. The invention is therefore not limited by the description contained herein or by the drawings, but only by the claims.

What is claimed:

1. An article of manufacture for use in a system that includes a display; a processor for controlling the display; and display object data the processor can use to generate first and second display objects the processor can present on the display; the article comprising:

memory that can be accessed by the processor; and

data stored in the memory; the data comprising:

first and second workspace data structures relating respectively to first and second workspaces that the processor can present on the display; each of the first and second workspaces including a respective set of display objects; each of the display objects being perceptible as a distinct, coherent set of display features; the display objects of each respective set being perceptible as having spatial positions relative to each other when the respective workspace is presented on the display; and

first and second linking data structures; the first linking data structure linking the first workspace data structure and the display object data so that the processor presents the first display object in the first workspace's set of display objects; the second linking data structure linking the second workspace data structure and the display object data so that the processor presents the second display object in the second workspace's set of display objects; the first and second display objects being perceptible as the same tool.

2. The article of manufacture of claim 1 in which the first linking data structure includes first display characteristic data indicating display characteristics of the first display object when presented in the first workspace's set of display objects.

3. The article of manufacture of claim 2 in which the second linking data structure includes second display characteristic data indicating display characteristics of the second display object when presented in the second workspace's set of display objects, the first and second display characteristic data being different from each other so that the first display object is presented differently in the first workspace's set of display objects than the second display object is presented in the second workspace's set of display objects.

4. The article of manufacture of claim 2 in which the first display characteristic data includes position data indicating a display position of the first display object.

5. The article of manufacture of claim 2 in which the first display characteristic data includes dimension data indicating a display dimension of the first display object.

6. The article of manufacture of claim 1, further comprising:

a third workspace data structure relating to a third workspaces that the processor can present on the display; the third workspace including a respective set of display objects; each of the display objects being perceptible as a distinct, coherent set of display features; the display objects of each respective set being perceptible as having spatial positions relative to each other when the third workspace is presented on the display; and

a third linking data structure linking the third workspace data structure and the display object data, the first and second linking data structures each comprising data identifying the third workspace data structure so that the third workspace is presented when one of the first and second workspaces is presented.

7. The article of manufacture of claim 1 in which the display object data may be accessed with a handle, the first and second linking data structures each including the handle.

8. An article of manufacture for use in a system that includes a display, a processor for controlling the display; and display object data the processor can use to generate first and second display objects the processor can present on the display; the article comprising:

memory that can be accessed by the processor; and

data stored in the memory; the data comprising workspace data the processor can use to present first and second workspaces on the display; each of the first and second workspaces including a respective set

of display objects; each of the display objects being perceptible as a distinct, coherent set of display features; the display objects of each respective set being perceptible as having spatial positions relative to each other when the respective workspace is presented on the display;

the processor presenting the first display object in the first workspace's set of display objects; the processor presenting the second display object in the second workspace's set of display objects; the first and second display objects being perceptible as the same tool.

9. The article of manufacture of claim 8 in which the display object data include a preexisting display system that provides windows, the first and second display objects being windows.

10. The article of manufacture of claim 9 in which the first and second display objects have the same receiving and response procedures from the preexisting display system.

11. The article of manufacture of claim 8 in which the display object data include a preexisting display system that provides windows and icons, the first display object being a window and the second display object being an icon.

12. The article of manufacture of claim 8 in which the workspace data comprise a special workspace data structure comprising data the processor can use in presenting a special workspace that is presented when any other workspace is presented; the special workspace including the first display object when the first workspace is presented; the special workspace including the second display object when the second workspace is presented.

13. The article of manufacture of claim 12 in which the workspace data further comprise workspace leaving procedure data indicating a procedure the processor can perform in leaving the first workspace to update a display characteristic of the first display object in the special workspace.

14. The article of manufacture of claim 13 in which the workspace data further comprise workspace entering procedure data indicating a procedure the processor can perform in entering the second workspace after leaving the first workspace to present the second display object in the special workspace with the updated display characteristic.

15. The article of manufacture of claim 8 in which the workspace data comprise:

workspace leaving procedure data indicating a procedure the processor can perform in leaving the first workspace to obtain a signal from a user indicating that the first display object should be moved or copied into the second workspace; and

workspace entering procedure data indicating a procedure the processor can perform in entering the second workspace to present the second display object in the second workspace's set of display objects.

16. The article of manufacture of claim 8 in which the workspace data comprise:

a first workspace data structure comprising data the processor can use in presenting the first workspace; and

a second workspace data structure comprising data the processor can use in presenting the second workspace.

17. The article of manufacture of claim 16 in which the first workspace data structure comprises a first linking data structure linking the first workspace data structure to the display object data so that the processor presents the first display object in the first workspace's set of display objects; the second workspace data structure comprising a second linking data structure linking the second workspace data structure to the display object data so that the processor presents the second display object in the second workspace's set of display objects.

18. The article of manufacture of claim 16 in which the workspace data further comprise:

a third workspace data structure comprising data the processor can use in presenting a third workspace on the display; the third workspace including a respective set of display objects; each of the display objects being perceptible as a distinct, coherent set of display features; the display objects of the respective set being perceptible as having spatial positions relative to each other when the third workspace is presented on the display;

a first linking data structure linking the first workspace data structure to the third workspace data structure so that the third workspace is presented on the display when the first workspace is presented on the display;

a second linking data structure linking the second workspace data structure to the third workspace data structure so that the third workspace is presented on the display when the second workspace is presented on the display; and

a third linking data structure linking the third workspace data structure to the display object data so that the processor presents the first display object in the first workspace's set of display objects when the first workspace is presented on the display and so that the processor presents the second display object in the second workspace's set of display objects when the second workspace is presented on the display.

19. The article of manufacture of claim 18 in which the third workspace data structure can be accessed with a handle; the first linking data structure and the second linking data structure each comprising handle data indicating the handle of the third workspace data structure.

20. The article of manufacture of claim 18 in which the first linking data structure comprises data indicating a display characteristic of the third workspace.

21. The article of manufacture of claim 16 in which the first workspace data structure comprises data indicating a procedure the processor can perform in leaving the first workspace.

22. The article of manufacture of claim 16 in which the first workspace data structure comprises data indicating a procedure the processor can perform in entering the first workspace.

23. The article of manufacture of claim 8 in which the workspace data further comprise transfer operation data indicating a procedure the processor can perform so that the first and second display objects are displayed at the same position and size in the first and second workspaces.

24. An article of manufacture for use in a system that includes a display; a processor for controlling the display; and display object data the processor can use to generate display objects the processor can present on the display; the article comprising:

memory that can be accessed by the processor; and

data stored in the memory; the data comprising:

workspace data the processor can use to present a plurality of workspaces on the display; each of the workspaces including a respective set of display objects; each of the display objects being perceptible as a distinct, coherent set of display features; the display objects of each respective set being perceptible as having spatial positions relative to each other when the respective workspace is presented on the display; the workspace data being linked to the display object data so that each display object in a workspace's respective set of display objects is included in the workspace when the workspace is presented on the display; the workspaces including a first workspace; the respective set of display objects of the first workspace including a first display object; the first display object being presented within the first workspace at full size when the first workspace is presented on the display; and

procedure data defining a procedure the processor can perform in accessing the workspace data structures and presenting representations of all of the workspaces simultaneously on the display, each workspace representation comprising a small-size representation of each display object in the workspace's respective set of display objects, the representation of the first workspace including a respective small-size representation of the first display object.

25. The article of manufacture of claim **24** in which the display object data include an application the processor can call in generating the first display object; the procedure data defining a procedure that presents the representations of all of the workspaces with the small-size representation of the first display object including a label identifying the application.

26. The article of manufacture of claim **24** in which the display object data include a preexisting display system; in performing the procedure, the processor determining a type of the first display object and presenting the representations of all of the workspaces with the small-size representation of the first display object including information to be displayed for the type.

27. A method of operating a system that includes:

a display;

input circuitry that receives signals from a user;

a processor connected to receive signals from the input circuitry and to control presentation of images on the display;

display object data the processor can use to generate display objects the processor can present on the display; and

workspace data the processor can use to present any of a number of workspaces on the display, the number of workspaces being two or greater; each of the workspaces including a respective set of display objects; each of the display objects being perceptible as a distinct, coherent set of display features; the display objects of each workspace's set being perceptible as having spatial positions relative to each other when the workspace is presented on the display;

the method comprising:

using the workspace data and the display object data to present a first one of the workspaces; the first workspace including a first one of the display objects at a first position and a first size;

receiving a first signal from the input circuitry, the first signal indicating that the first display object is to be presented at the same position and size in every one of the workspaces;

receiving a second signal from the input circuitry, the second signal requesting a switch from the first workspace to a second one of the workspaces; and

using the workspace data and the display object data to present the second workspace; the second workspace including a second one of the display objects at a second position that is the same as the first position and at a second size that is the same as the first size; the first and second display objects being perceptible as the same tool.

* * * * *