EDITABLE GRAPHICAL HISTORIES: THE VIDEO

David Kurlander
Steven Feiner

Department of Computer Science
Columbia University
New York, NY 10027

INTRODUCTION

Command histories are an important component of good user interfaces [3]. They allow users to review sequences of previously executed commands, and can provide an interface to an undo facility. Furthermore, they can support either a simple redo mechanism, or a more sophisticated macro-by-demonstration capability. Providing a visual representation of command histories in a graphical user interface presents a number of difficulties. Editable graphical histories [1, 2], a history representation that we have developed based on a comic-strip metaphor, overcomes many of these. Here we describe editable graphical histories, and in the accompanying videotape we demonstrate our test-bed implementation in the graphical editing mode of Chimera, a multi-modal editor.

DESCRIPTION

The most popular history representation for text-based applications is the *virtual scroll* model, where the command lines executed are displayed one after another in a list. Graphical user interfaces are becoming increasingly popular today, but it is difficult to represent their command histories in the same fashion, since the arguments to these commands may be graphical objects whose screen positions are critical to their interpretation. Often textual command names are forgotten when mouse and keyboard accelerators are used. As a result, graphical interfaces often abandon textual command histories altogether, and instead present snapshots of the application.

One such approach to presenting history is *temporal reexecution*, in which the application can be scrolled forwards and backwards in time, as the visual side effects of the commands are observed. Working with this representation is akin to editing a document with a line editor—only a narrow slice can be viewed at once, and the user must keep a mental model of the rest. Temporal reexecution depicts the effects of commands, and a virtual scroll often represents only the commands themselves. Neither structures the history presentation to facilitate understanding.

Editable graphical histories merge the spatial and temporal paradigms, to produce a history representation for graphical user interfaces that avoids many of the disadvantages of traditional representations. We rely on the metaphor of a comic strip, showing, in a series of pictorial panels, the important events in the history of an application.

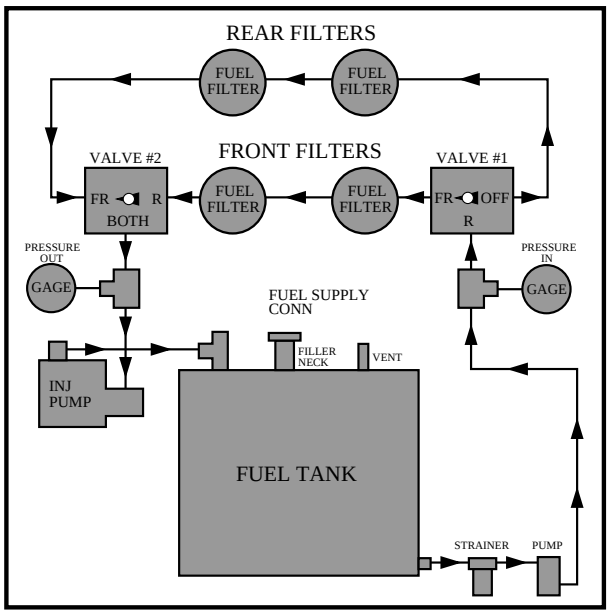
If each event were to be allocated its own panel, then the history would be difficult to read, since the dialogue would be at an extremely low level, and there would be a proliferation of panels. Instead, we rely on an *inter-panel detail removal* mechanism, to coalesce multiple related actions into single panels. Our system contains a grammar that parses multiple primitive commands into higher-level actions. A user-settable *granularity level* adjusts how much coalescing is performed. If it becomes necessary to view the operations composing a history panel in more detail, an *interactive elaboration* mechanism allows higher-level panels to be decomposed into lower-level ones.

If each panel were a complete miniature of the application window, then the small size would obscure the effects of the operations. Instead of using either a screen miniature or fixed icon, we rely on an *intra-panel content selection* mechanism to choose the important objects to depict in each panel. We also render objects in a graphical style that emphasizes their roles in the explanation.

EXAMPLE

An example of an editor scene and several panels from its history are shown in Figure 1. The history panels are generated in vertical pairs, showing the application before and after important operations. Using before-and-after pairs allows us to avoid relying on symbols for expressing the history that are not used in the interface. The top panel (the *prologue*) depicts the arguments of the commands as they appeared before the important operation was performed. The effects of the operation are shown in the bottom panel (the *epilogue*). Related operations are coalesced into individual panel pairs.

(a)



(b)

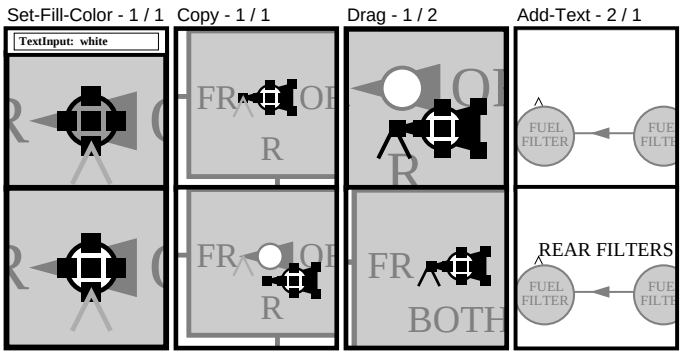


Figure 1. Editor scene (a), and four frames from its graphical history (b).

In the first panel pair of Figure 1b, we set the fill color of the circle to be white. The arguments of the operation, the color typed into the Text Input widget of Chimera's control panel, and the selected circle of the editor scene, are shown in the prologue. The modified circle appears in the epilogue. In order to disambiguate the location of the graphical objects, the history mechanism includes other nearby objects in the panels as well. However, these contextual objects are deemphasized by lightening their colors.

The first two panel pairs each depict two operations: both an object selection, and a more significant operation that is labeled above the panels to further disambiguate the command. In the second pair, the selection is extended to include the triangle, and the selections are copied. The third panel pair represents the copy being dragged to a new position in the Valve #2 box, visible in Figure 1a. This pair represents three operations coalesced together—a Move-Caret operation (which positions the software cursor), and two Drag operations. These two Drag operations appear in the same panel, since multiple translations of a single set of objects is equivalent to a single translation. In the last panel pair, a text label is added to the scene. Two caret movements immediately prior to this operation are also coalesced into this pair, since the text is added at the caret location, and multiple caret movements are equivalent to a single one. These panels can be broken up into additional lower-level panels if further detail is needed.

CONCLUSION

Editable graphical histories are currently implemented as part of the Chimera editor, which we use to experiment with user interface issues. These histories form a critical component of Chimera's undo interface. By selecting a panel with the mouse, the user can revert the editor scene back to any point in time depicted. After subsequent user interaction, an undone sequence of operations can be redone, and the history is automatically updated to show the new command sequence.

We are in the process of moving the history mechanism out of Chimera and into our user interface management system, so that other applications can take advantage of it as well. In addition, we are building a macro-by-demonstration facility which will use histories both as a means of selecting sequences of operations to be encapsulated into a macro, and as part of the visual representation of the macro itself.

REFERENCES

1. Kurlander, D. and Feiner, S. Editable Graphical Histories. *Proc. 1988 IEEE Workshop on Visual Languages.* (Pittsburgh, Oct. 10–12, 1988). 127–134. Reprinted in *Visual Programming Environments: Applications and Issues*, E. P. Glinert, ed. IEEE Press, Los Alamitos, CA, 1990. 416–423.

2. Kurlander, D. and Feiner, S. A Visual Language for Browsing, Undoing, and Redoing Graphical Interface Commands. In *Visual Languages and Visual Programming*, Shi-Kuo Chang, ed. Plenum Press, New York, 1990. 257–275.

3. Lee, A. A Taxonomy of Uses of Interaction History. *Proc. Graphics Interface '90.* (Halifax, May 14–18, 1990). 113–122.