1995 IEEE Symposium on Visual Languages, Darmstadt, Germany, Sept. 5-9, 1995

Integrating Algorithm Animation into a Declarative Visual Programming Language

Paul Carlson and Margaret M. Burnett
Department of Computer Science, Oregon State University
Corvallis, Oregon 97331-3202 USA
E-mail: burnett@cs.orst.edu

Abstract

Until now, only users of textual programming languages have enjoyed the fruits of algorithm animation. Users of visual programming languages (VPLs) have been deprived of the unique semantic insights algorithm animation offers. To begin solving this shortcoming, we have seamlessly integrated algorithm animation capabilities into the VPL Forms/3. Our research shows how a declarative VPL that is responsive can provide features not found in other algorithm animation systems.

1: Introduction

This paper describes how we integrated a visual and declarative extension of an algorithm animation model called the path/transition paradigm [1] into Forms/3 [2], a declarative VPL. Our goal was not a system for animating algorithms implemented in other languages. Rather, our goal was to use a VPL to animate algorithms that are themselves programmed in that VPL.

At first we simply wanted to see if a declarative VPL could support algorithm animation without compromising the fundamental characteristics and concepts of that VPL. What we discovered was that not only are those characteristics and concepts not compromised, but that they can be exploited to produce an algorithm animation approach that features unusual characteristics.

2: Animation Programming in Forms/3

Forms/3 is a general-purpose VPL. Programming in Forms/3 follows the spreadsheet paradigm; the programmer uses direct manipulation to place cells on forms, and then defines a formula for each cell. Such a formula may include constants, references to other cells, or references to the cell's own value at a previous moment in time. Cells are referenced by clicking on them. (In our current prototype, these references are reflected textually in the formulas.) A program's calculations are entirely determined by these formulas.

Forms/3 is declarative and responsive. By declarative, we mean that programming is a matter of defining the relationships between the inputs and the desired outputs. By responsive, we mean that whenever an incremental change in a program is made, its effects are immediately and automatically reflected in the displayed portion of the program's results.

Figure 1 shows an animation of an in-place selection sort. The sort swaps the minimum of the remaining unsorted elements with the element in the first unsorted position. Such a sort is possible in the declarative Forms/3 due to the language's inclusion of an explicit time dimension. Figure 2 shows the user's view of the sort itself. The list to be sorted is defined by the user in the initialList cell. The in-place sort occurs over time in the group of cells in the lower-left portion of the form. The algorithm does not know about its animation.

The programmer constructed this animation by customizing Forms/3's built-in Animation form using one position in the list as a test case, and then passing the other positions of the list to other copies of that form. The programmer's only other task was specifying the formula shown in Figure 1.

Figure 3 shows the customized Animation form. The programmer customized the form in two ways: by adding two new cells (value and whichPosition, upper left corner) to better communicate how this animation works, and by replacing default values with the new formulas shown.

3: Discussion

Forms/3's declarative and responsive characteristics produced features that are unusual in algorithm animation.

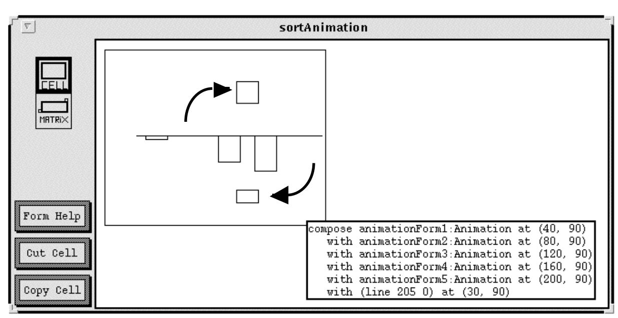

Figure 1: Animation of an in-place selection sort. The arrows are superimposed on the screen shot to show the direction of motion.

On-the-fly exploratory programming: Because of the combination of responsiveness with declarativeness, algorithm animation programming in Forms/3 does not have the edit-compile-restart loop of traditional algorithm animation programming. Instead, a programmer can modify an animation or even the animated algorithm itself *during mid-execution* (a capability of growing importance in scientific computing known as 'steering'), and receive immediate feedback about the effects of those modifications. Such a change in Forms/3 redefines the complete history of the algorithm and its animation, and the screen display is updated to reflect the point in execution at which the modification was added.

Control over the direction of execution: Not only can the programmer modify the animation (or algorithm) during runtime for exploratory or debugging purposes, but the changed program and its animation can be executed synchronously either forward or backward. The direction can be toggled at any time by simply clicking a button in the programming environment, permitting the programmer to step forward and backward at will, examining any portion of the execution as many times as desired without restarting the program.

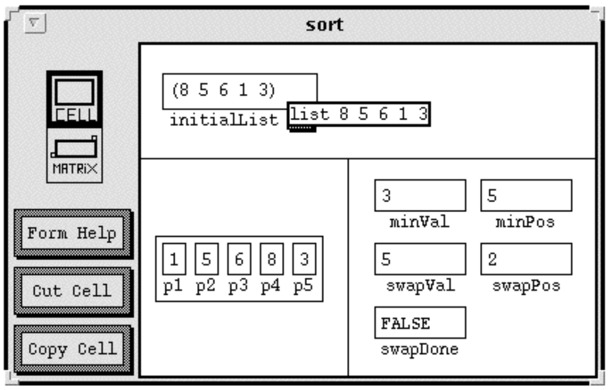

Figure 2: The in-place selection sort during execution.

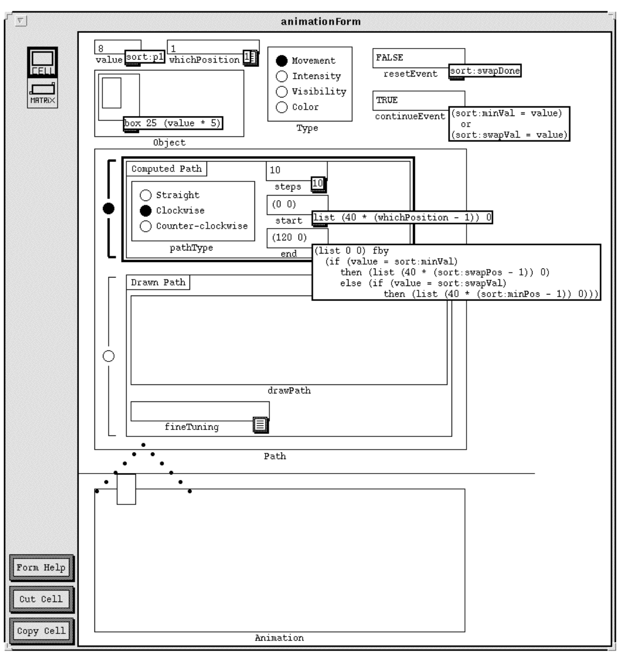

Figure 3: A representative Animation form from the in-place selection sort animation.

Independence of the algorithm from the animation: In Forms/3, as in other declarative animation systems, animation programming can be thought of as defining a mapping from computational states to a desired animation. The advantage of this declarative mapping approach is that the algorithm being animated is neither altered nor augmented with animation code. Instead, the interesting events of the algorithm are defined in the *animation* code. This allows the same algorithm program to be used both with and without its animation without the need for separate compiled (or source) versions of the program.

Generalized event-based approach to advancing animation Our approach uses events to advance, as well as to activate, animation. After all, the traditional mechanism of using time to progress from one animation frame to the next is really just the occurrence of a specialized event—the passage of a time interval—which can be replaced by the occurrence of any general event. The programmer defines this general event—which can be time-based if desired—in the formula of the continueEvent cell on the Animation form.

4: Conclusion

This work shows how a VPL that features responsiveness and declarativeness can be extended to produce unusual features for algorithm animation. A more detailed description of the approach and of an earlier version of its implementation can be found in [3].

References

[1] J. Stasko, "The Path-Transition Paradigm: A Practical Methodology for Adding Animation to Program Interfaces," *J. Vis. Lang. Computing*, 213-236, Sept. 1990.

[2] M. Burnett, A. Ambler, "Interactive Visual Data Abstraction in a Declarative Visual Programming Language," *J. Vis. Lang. Computing*, 29-60, Mar. 1994.

[3] P. Carlson, "Algorithm Animation in a Declarative Visual Programming Language," M.S. Thesis, Oregon State Univ., Computer Science, 1995. TR 95-60-2, Apr. 1995.