Designing Pedagogical Screen Savers

Chris DiGiano and Michael Eisenberg
Center for LifeLong Learning and Design
Department of Computer Science and Institute of Cognitive Science
University of Colorado at Boulder
Campus Box 430, Boulder, Colorado 80309-0430
tel: (303) 492-1218, email: {digi,duck}@cs.colorado.edu

ABSTRACT
The burgeoning complexity of professional application software—the proliferation of interface options, available functionality, and end-user languages—has resulted in the need to think creatively about ways in which such software may be made more learnable. This paper describes one promising technique—the *pedagogical screen saver*—whose purpose is to introduce users to application functionality, entertainingly and unobtrusively, during the program's "idle time." We describe a running prototype of such a screen saver for a programmable charting application.

KEYWORDS
Screen savers, programmable applications, software design, Chart 'n' Art.

INTRODUCTION
It is by now a commonplace observation that professional software applications are becoming increasingly complex, through the continual accretion of features, interface options, and (in the case of programmable applications) end user language vocabulary. While it is certainly arguable that some of this increased complexity is unnecessary (and even counterproductive) [2], there is little question but that the "learnability problem" for applications will not go away. As a result, designers of applications will need to think ever more creatively about ways in which users may be introduced to the range of features and concepts available in their professional tools. This paper describes a technique that we believe will prove promising as an aid to learnability: the *pedagogical screen saver*. In the remainder of this section, we outline the basic design principles behind the construction of pedagogical screen savers; in the second section we present a running prototype of the idea, as implemented for a programmable charting application, *Chart 'n' Art*.

The essential motivation behind the pedagogical screen saver is that, for application users, much learning *could* in principle take place by exploration—but, as observed by Rieman [4] in his (admittedly small-scale) study, relatively few users engage in active exploration of their software, and when users do engage in exploration it is typically motivated by specific work-related tasks. Ideally, then, we would like a mechanism that permits users to learn "serendipitously," in idle moments, about software functionality, much as in the "did-you-know" mechanism described by Owen [3]. At the same time, however, the information presented in this serendipitous fashion should be, as much as possible, related to the user's current work, goals, or interests.

The pedagogical screen saver is intended to respect this style of "goal-driven-but-serendipitous" exploration. The basic mechanism is one in which a given application is associated with its own screen saver; given a sufficiently long period of program inactivity, this screen saver will present, in animated fashion, interesting or novel features of the application. In the very simplest version of the idea, these features could be presented as "canned" examples; this would be reminiscent of Owen's system or the way in which arcade video games present instructions and hints between actual games. A more interesting and probably more useful version of the idea—following from the observations of the previous paragraph—would stipulate that the examples presented should be based on recent activity of the user. In this case the screen saver needs to take into account some history of the application's recent activity, presenting examples that expand on the user's own projects.

Our own belief is that the notion of a pedagogical screen saver will eventually prove most useful when integrated with end user programming; here, the purpose of the screen saver would be to introduce the user to the vocabulary or programming constructs of an application's associated language. In this sense, pedagogical screen savers may be seen as a natural extension of other means for unobtrusively introducing language features to users through techniques that we have broadly described as *self-disclosure*. Very briefly, a self-disclosing system is one that permits application users to work with direct manipulation constructs; as those constructs are employed, the application informs the user of related or equivalent ideas within the application's language [1]. The ideas behind the pedagogical screen saver are natural outgrowths of the general principles of self-disclosure: that is, the screen saver should work subtly and unobtrusively, introducing users to language constructs most directly related to their own current or recent activity.

A PEDAGOGICAL SCREEN SAVER FOR A PROGRAMMABLE CHARTING APPLICATION

Chart 'n' Art is a programmable application for the creation of novel charts and information displays; the system is described at greater length in [1], but here we focus on its pedagogical screen saver. The screen saver is activated after the application has been idle for some time. At this point, the system changes the screen's background color to black and hides all but two windows: the current drawing being edited, and the "Language Ideas" window used in the running application for self-disclosure. To guarantee the integrity of the original work, the system creates copies of the current drawing and Language Ideas windows. After hiding the original windows, the screen saver proceeds to alter the drawing at random, making one small change every five seconds as shown in Figure 1. Possible changes made by the screen saver include adjusting properties of existing shapes such as their size, color, and screen position, as well as creating entirely new shapes.

The screen saver continues to evolve the drawing in this way until making some 15 changes. This usually renders a quite different picture, but one that still resembles the original construction. At this point the system momentarily turns the screen to black (to assure that it is indeed "saving" the screen), and then starts altering two fresh copies of the original drawing and Language Ideas windows. The entire process repeats itself until the user moves the mouse or touches the keyboard.

Although the screen saver's randomly evolving drawing can be engaging in and of itself, the system's most unique and educational aspect is reflected in the language disclosures that accompany each change as shown in the text windows in Figure 1. These disclosures are programming language expressions users could have typed to achieve the same effects themselves. Such feedback provides information on how users could employ the system's end-user language to modify and extend their existing constructions. The actual modifications shown by the screen saver are chosen at random from a list of legal commands generated automatically from a language definition file. Using this general mechanism, Chart 'n' Art's screen saver can present any number of end-user languages. Currently, the system supports either SK8Script [5] (as shown in Figure 1) or Lisp.

Changes made by Chart 'n' Art's screen saver, although unlikely to actually improve upon the original artifact, provide users with an idea of the space of possibilities available to them through direct manipulation and programming language operations. One can imagine how such feedback could be particularly useful for complex, high-functional systems such as PhotoShop[1] or AutoCAD[2] featuring literally hundreds of transformations and effects.

[1] PhotoShop is a registered trademark of Adobe Systems, Inc.

[2] AutoCAD is a registered trademark of AutoDesk Corporation.

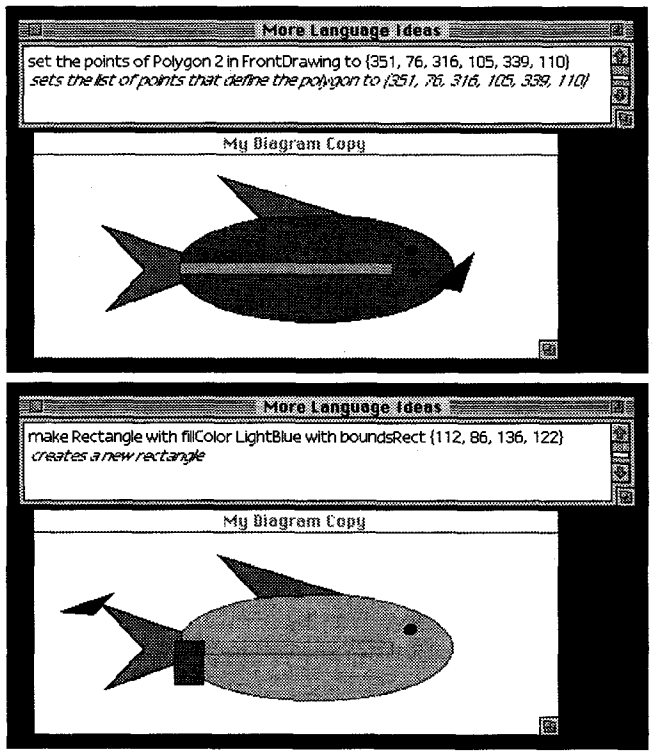

Figure 1: A diagram being modified by Chart 'n' Art's screen saver. The top frame shows the screen saver's first change—reshaping the fish's polygonal mouth—and its corresponding language disclosure. The bottom frame shows the same construction several alterations later, when the screen saver introduces a random new shape and discloses the "make Rectangle" command.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation grant RED-9253425 and a Young Investigator award IRI-9258684., as well as NSF-Advanced Research Projects Agency Cooperative Agreement CDA-940860.

REFERENCES

1. DiGiano, C. and Eisenberg, M. Self-Disclosing Design Tools: A gentle introduction to end-user programming. In Proc. DIS '95, ACM Press, Ann Arbor, 1995, pp. 189-197.

2. Eisenberg, M. and Fischer, G. Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance. In Proc. CHI'94, ACM Press, New York, 1994, pp. 431-437.

3. Owen, D. Answers First, Then Questions. In User Centered System Design, New Perspectives on Human-Computer Interaction, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 361-375.

4. Rieman, J. Field Study of Exploratory Learning Strategies. ACM Transactions on Computer-Human Interaction, (in press).

5. Spohrer, J. Apple Computer's Authoring Tools and Titles R&D Program. AI Review, 9 (1995), pp. 85-89.