Shrager, J., & Finin, T. (1982). An expert system that volunteers advice. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 74-78). Los Altos, CA: Morgan Kaufman.

About the authors...

The authors made up the Intelligent User Assistance team in the MCC Human Interface Laboratory. Research interests of the team include interface design, computer-mediated communication and collaboration, advisory and tutorial systems, applications of artificial intelligence techniques to user interfaces, and computers in the arts and humanities. All the authors can be contacted at:

MCC Human Interface Laboratory
3500 West Balcones Center Drive
Austin, Texas 78759

except for Miller, who is now at:

Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, California 94304

TANGO: A FRAMEWORK AND SYSTEM FOR ALGORITHM ANIMATION

JOHN T. STASKO

Algorithm animation is the process of abstracting the data, operations, and semantics of computer programs, and then creating animated graphical views of those abstractions. Although a handful of algorithm animation systems have been developed in recent years, relatively little work has been done on the theoretical foundations of such systems. In this work we develop a conceptual framework with formal models and precise semantics for algorithm animation. The framework contains facilities for defining operations in an algorithm, designing animations, and mapping the algorithm operations to their corresponding animations. Concurrently, we develop an algorithm animation system called TANGO (Transition-based ANimation Geverati On) and a WYSIWYG demonstrational animation design tool called DANCE (Demonstrational ANimation CrEation) that are both based upon the framework.

Four primary goals strongly influenced our framework for algorithm animation:

- The framework should provide a model of what an algorithm animation is, and how the images in an animation undergo changes and modifications throughout the animation's frames.
- The animation framework should be user-oriented; animation development time should not outweigh the time used to design the algorithm or program being animated. Previous systems have often been complex enough so that only the system's developer was able to produce new animations. Our system emphasizes a simplified, consistent design paradigm that allows programmers to create their own animations in a relatively short time.
- The animation framework should provide a model that supports smooth, continuous transitions and movements of graphical images in an animation. These transitions are preferable to discrete graphical updates that appear in a snap shot-like sequence and that are often aesthetically unpleasant.
- The animation framework should support a system that functions in a widely available teaching and research environment: the C programming language and the UNIX operating system.

Animations within the framework are based upon a *path-transition* paradigm: in the viewing window, graphical images undergo transitions such as movement and alteration of size or color along two-dimensional paths. The animation framework itself is based upon four algorithm animation data types: locations, images, paths, and transitions. *Locations* identify particular graphical positions of interest that can be saved and used as markers. *Images* are the simple visual pictures that are modified to produce animations. *Paths* are sequences of two-dimensional control points along which image modification occurs. The most common use of a path is to provide a trajectory to move an image along, but paths also define the way an image changes size, color, visibility, and so forth. *Transitions* are the logical units of action or change in the animations. They are defined by a transition type such as movement or coloration, the image affected, and a path along which to modify. A powerful set of operations upon the four data types allows objects to be created, edited, saved, and reused.

The framework supports two primary mappings from a program into its animation. Mapping program data to animation data type objects is performed through the use of *associations*, methods to store and retrieve data. For example, a sorting program's animation may contain an association from individual array elements in the program to their corresponding picture images in the animation window. The second mapping, from

abstract operations in the program to the corresponding animation actions, is performed through a versatile finite state automaton model that provides the animation designer with sophisticated animation control.

The TANGO algorithm animation system that we have developed is based upon this conceptual framework. TANGO supports two-dimensional color animations in a window-based workstation environment. To drive TANGO animations, programmers supplement their algorithms with events that we call algorithm operations (the BALSA approach). As program execution occurs, events are sent out as interprocess messages to a central message server that passes the messages on to TANGO animation windows. Individual messages generate state transitions within TANGO and activate the appropriate animation routines. These animation routines are user-written procedures in our algorithm animation design language that consists of the four animation data types, their framework operations, and the association calls, all embedded within C.

To aid in the development of the animation routines, we have created a WYSIWYG demonstrational tool called DANCE that further simplifies animation design in TANGO. DANCE provides MacDraw-like graphical design capabilities supplemented by animation transitions. Using DANCE, programmers graphically demonstrate the actions to occur in an animation, and then DANCE generates the algorithm animation design language code that carries out those actions. DANCE produces code that compiles and executes unaltered, but that still can be edited and "tweaked" when desired.

TANGO currently runs on a network of SUN and DEC workstations. It uses tools from the Brown Workstation Environment (BWE) user interface toolkit and the FIELD programming environment. Both are X11-based. We have used TANGO to animate programs from a wide variety of domains such as sorting, searching, hashing, and graph and tree manipulations. We have also designed animated simulations of a producer-consumer ring buffer, the Towers of Hanoi problem, and the post office queuing problem. Students from a computational geometry course at Brown have used TANGO to visualize algorithms in that domain. Both BWE and FIELD are available for distribution. TANGO will soon be included within the FIELD distribution.

References:

John T. Stasko. *TANGO: A Framework and System for Algorithm Animation.* PhD Thesis, Brown University, Providence, RI, 1989. Available as technical report CS-89-30.

Steven P. Reiss and John T. Stasko. The Brown Workstation Environment: A User Interface Design Toolkit. In *Proc. of the IFIP Working Conference on Engineering for Human-Computer Interaction*, Napa Valley, CA, August 1989.

Steven P. Reiss. Interacting with the FIELD Environment. Brown University, technical report in preparation.

About the Author

John T. Stasko recently received his Ph.D. in computer science from Brown University in Providence, RI. This article is a brief summary of that work. Stasko is now an assistant professor at Georgia Tech. He can be reached at

Dept. of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332.

stasko@prism.gatech.edu

DESIGNING COLLABORATIVE USER INTERFACES: LESSONS FROM WRITER/GRAPHIC DESIGNER INTERACTION

WENDIE WULFF

Introduction

Recent discussions about designing and building user interfaces for collaborative work have revealed a number of points. One of the most significant is that before we can concentrate on collaborative interfaces, we still have much to learn about the nature of collaboration itself. In particular, we need information from studies that have systematically looked at interdisciplinary collaboration in real contexts, with real people and problems. Thus, the goals of my research are to deeply understand one kind of interdisciplinary collaborative work -- the activities of a document design team working on a client project -- and then to produce guidelines for computer support of the interactions studied.

What is document design?

Document design is the practice and study of the creation and reception of functional visual and verbal texts -- texts intended to inform, explain and instruct.

Document designers solve communication problems for clients who have messages to convey, products to describe, or procedures to present. Document designers also solve communication problems for readers by making