

**One Step Towards Multimedia  
Help: Adding Context-Sensitive  
Animated Help as One Ingredient**

by

**Piyawadee "Noi" Sukaviriya**

**GIT-GVU-91-19  
October 1991**

**Graphics, Visualization & Usability  
Center**

**Georgia Institute of Technology  
Atlanta GA 30332-0280**

# ONE STEP TOWARDS MULTIMEDIA HELP: ADDING CONTEXT-SENSITIVE ANIMATED HELP AS ONE INGREDIENT

Piyawadee "Noi" Sukaviriya

College of Computing  
Georgia Institute of Technology  
801 Atlantic Dr., Atlanta, GA 30332-0280  
Phone: (404) 894-9105  
E-mail: noi@cc.gatech.edu

## ABSTRACT

Users often have difficulties relating general help information to the specific computer tasks which they are attempting to complete. A factor contributing to this problem is the distance between the space in which help is presented and the space in which users have to apply what they learn from help. Context-sensitive multimedia help potentially can reduce these distances. This paper discusses the use of *context-sensitive graphical animation* as one medium of help presentation in which the user's task context is synthesized as part of procedural help demonstrations. This paper also discusses how a support for this kind of help has been integrated in an application environment, thus enables an automatic generation of this kind of help for various applications.

**KEYWORDS:** Animated Help, Procedural Help, Context-sensitive Help, User Interface Environment, Multimedia Help

## INTRODUCTION

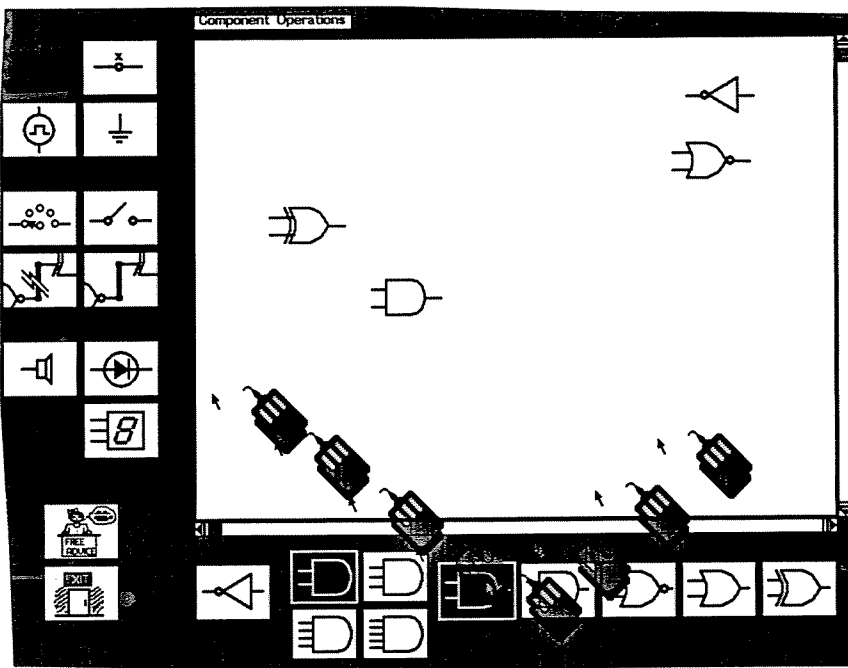
Traditional help contains general information which is independent of the context in which help is requested. Users often have difficulties relating general help information to the specific computer tasks which they are attempting to complete. A major factor contributing to this problem is the *conceptual* distance between the help space and the computer task problem space. Failure to relate information obtained from help to current problems could discourage users from consulting on-line help the next time a new problem occurs. Help would be more useful if it presented information in such a way that its content were directly related to the context of the user's question.

Another factor contributing to the relevancy of help information problem is the *representational* distance between the help interface space—the space in which help is presented, and the application user interface space—the space to which users apply what they learn from help. In a

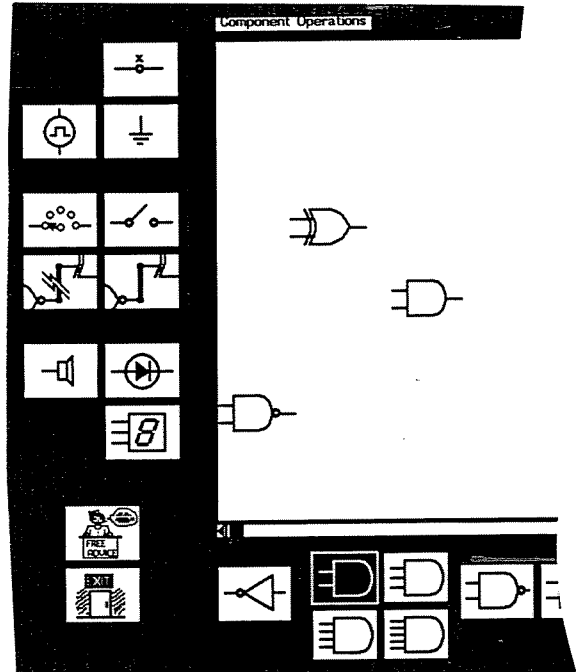
traditional textual interface with textual help, this "representational" distance barely exists since both help and a user's problem reside in the same textual space. With the introduction of advanced user interfaces employing more graphics, and now multimedia, the distance juxtaposed with older text-oriented help technology has been stretched. Using traditional textual help in advanced interfaces imposes an additional cognitive load on users because they must translate from the textual space to their actual work space.

Context-sensitive multimedia help potentially can reduce these distances. Introducing multimedia technology to help implies that help materials can possibly be presented in a way which is perceptually close to the application for which help is designed. This is more true in graphical interfaces where most tasks are not only visual but also animated in its nature. Dragging files, pulling down menus, moving the cursor from one place to another, etc., require continuous movements of input devices while the continuity of task context is maintained visually on the screen. To explain these processes in words as help to users is indirect. Users have to read and try to figure out where objects involved in a task are on the screen. Explaining which mouse buttons to use in a long task for a multiple-button mouse environment can be lengthy and distracting. Explanations of consequences of each action in a task in detail can be distractive, yet without them users do not foresee the consequences and are often surprised and confused when they perform the task. When a task description is long, an attempt to learn and memorize from textual help what to do in an actual graphical interface often breaks down rapidly.

Visual media such as static graphics (representing snapshots of screen states), pre-recorded video of how to perform tasks, and "context-sensitive" animated help naturally encode some or most of the visual cues in performing task procedures. While static graphics and video images have to be pre-recorded and cannot be rendered specifically to any particular task situations, animated help can be tailored to the current user's context if designed properly. Making it even better as will be shown in the paper, its generation mechanism can be embedded in the application environment such that a current context is always accessible by help.

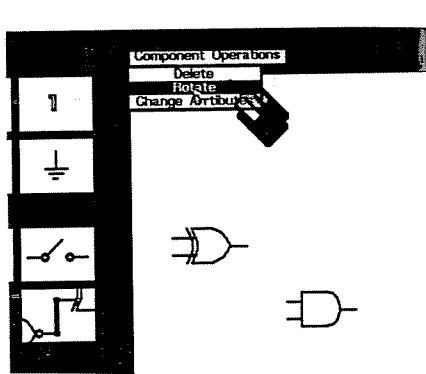


(a) Selecting the create NAND icon and selecting a location

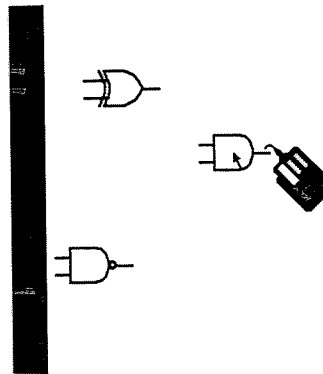


(b) A NAND gate is created as a result.

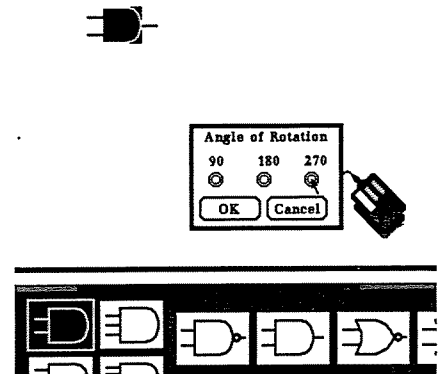
Figure 1 Animation of How to Create a NAND Gate



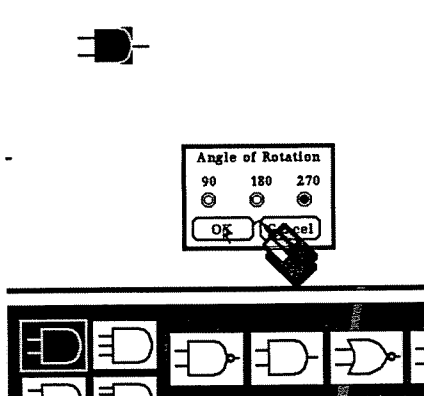
(a) Selecting ROTATE from the Pulldown menu



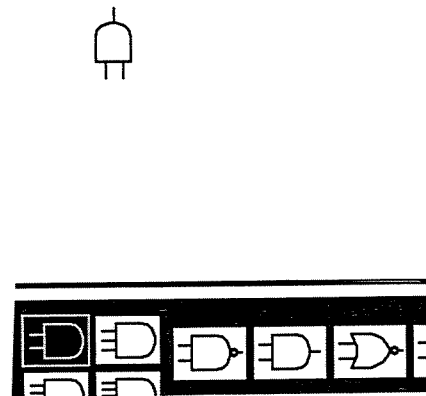
(b) Select a Gate



(c) Select a Degree of Rotation

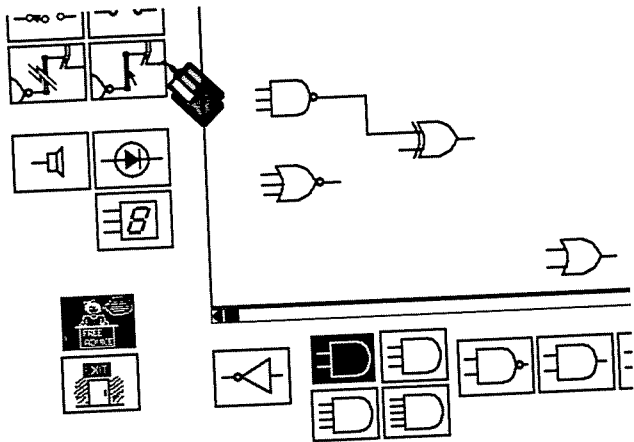


(c) Clicking on the OK Button

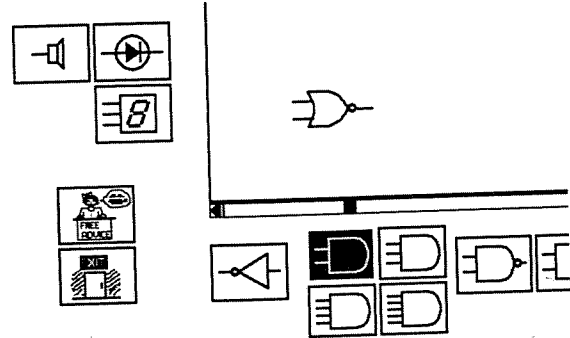


(d) The Selected Gate is then Rotated

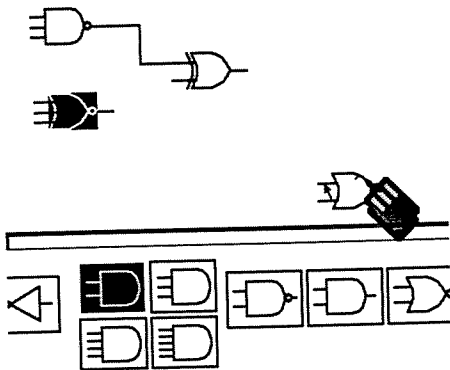
Figure 2 Animation of How to Rotate a Gate



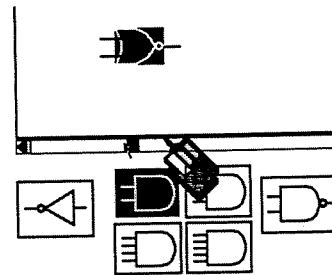
(a) Selecting the CONNECT Icon



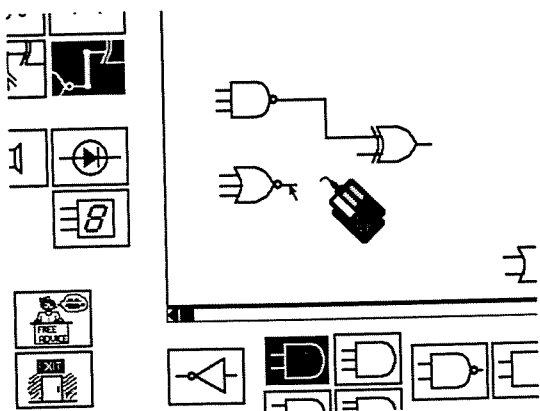
(a) The initial context: a gate is hidden in the left side of the window



(b) Selecting the Output Line of a Gate

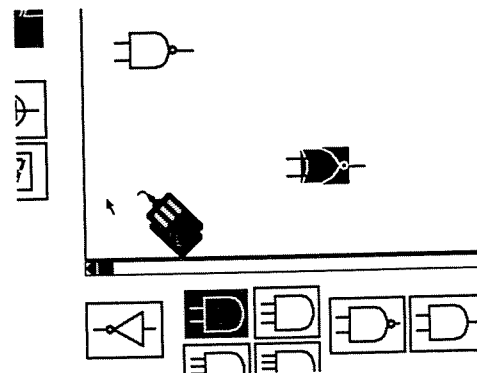


(b) Scrolling the Window to Bring the Hidden Gate Back



(c) Selecting an Input Line of Another Gate

Figure 3 Animation of How To Connect Two Gates



(c) Continuing with the Connect Animation

Figure 4 Animation of How to Connect When A Gate is Hidden

Each animation scene can then be constructed and played at runtime using existing objects, their current attribute values, and current screen states. For this reason, context-sensitive animated help is by far a medium which closes the representational distance between help presentation and the actual interface. Animation of procedures directly on an interface should be thought of as an alternative medium in a multimedia help environment.

In the following sections, we will show various examples of context-sensitive animated help captured from a system called *Cartoonist*, a runtime environment with embedded context-sensitive animated help generation mechanisms. We will then give a quick overview of the system, discuss the implication of this form of help to multimedia help, and summarize related work in the area of animated help.

In order to demonstrate context-sensitive animated help in the next section, a digital circuit layout program with a MacDraw-like interface is used as a sample application. In this application, a user creates and places different kinds of gates—AND, OR, NAND, NOR, NOT, XOR—in a drawing pad. Once a gate is created, it can be moved around in the pad or deleted. Its attributes, such as fan-in, fan-out, number of input and output lines, etc., can be modified. Created gates can be connected to form a composite digital circuit in the pad. (For a graphical interface, when two gates are connected, a wire which visually represents a physical link between the two gates is created.) Connected gates can also be disconnected (by deleting the wire which connects them). A number of other components such as those dealing with inputs and outputs can be created and manipulated as well. We made Figure 1 fully show the interface for this application.

#### WHAT IS CONTEXT-SENSITIVE ANIMATED HELP?

Animated help refers to procedural help, delivered as animated demonstrations of how to perform tasks within an application interface. As can be seen from Figures 1<sup>†</sup> and 2, animated help involves superimposing a graphical "moving" animated character on a graphical interface to provide a visual, dynamic presentation of operations (on input devices—the mouse in this case) which complete the procedure in question. The animated character represents an input device; its movement on the screen spatially corresponds to the physical movement of the represented device. Screen objects which can be used to complete the procedure in question are incorporated as part of the animation. Essentially, the animation combines simulated movement of input devices and existing objects within the visual space of the screen to directly suggest the means by which a task can be completed.

Context-sensitive help refers to help which takes into account the current context in which help is requested and tailors help presentations appropriate to the context.

---

<sup>†</sup> We left footprints of the mouse in this figure to help visualize the actual animation.

Context-sensitive animated help ideally would provide the kind of help a human expert would provide when giving help to a novice; when asked to help, the expert looks at the current user context, asks the users what they are trying to achieve, determines a reasonable solution for the given context, and shows the solution by explaining and demonstrating in that particular context. If the context needs to be rearranged so the procedure in question can be demonstrated, the expert would do so prior to demonstrating the intended procedure. Using the same paradigm, context-sensitive animated help not only animates requested procedural solutions, but also animates altering the current context so the task could be properly demonstrated.

Figures 3 a through c show a case where there are a number of gates already created in the layout; some of them are already connected, some are not. Assume that the user just loaded a design begun by another design team member and does not know how to connect gates at this point. Upon requesting help on *connect*, some of the gates are selected from the current context. In these figures, a straightforward animation of selecting the *connect* icon, selecting the output line of a gate, and an input line of another gate completes the *connect* procedure. Assume that the context before asking the system to help on how to *connect* is as shown in Figure 4a where there are actually two gates already created, but one gate is scrolled away. Notice the position of the elevator in the horizontal scroll bar at the bottom of the layout window. The hidden gate is scrolled away to the left of the window. Figures 4 b through c show a partial animation of how to *connect* at the point where the help system just finished showing selecting the *connect* icon and the output line of a gate (highlighted in both figures). The animation shows how to scroll the other gate back to the visible range, then continues with the *connect* procedure.

#### BEHIND ANIMATION SCENES

##### Environment

*Cartoonist* is a knowledge-driven user interface environment designed to meet the objective of supporting automatic generation of context-sensitive animated help. In *Cartoonist*, knowledge about an application and its interface specifications is captured at design time, to support the automatic runtime interface of the application including the context-sensitive animated help. An application designer defines all application actions to be supported in an application, then determines their bindings to interface actions. Once application routines are supplied for all actions and after some initial setup, *Cartoonist* is ready to run the application. The same knowledge is used to construct animations of how to perform tasks; the current context is synthesized for the animation scenes as part of the process.

*Cartoonist* is implemented in ParcPlace Smalltalk version 2.5. While the particular prototype for this research was mainly developed on Sun/3 and Sparc2 workstations, the ParcPlace Smalltalk platform is rather flexible and allows

the executable image of Cartoonist to be transferred to other machines that support the same version of Smalltalk without modifications.

### Knowledge Base

The Cartoonist's knowledge base, used as a basis to generate animation, is grouped in 3 categories: application action, interface action, and interaction techniques. The first two categories – application action and interface action – share the same representation. Application actions are those actions meaningful in a specific application domain. Examples are such as reducing the size of an image or drawing a centered heading on top of a page action in a desktop publishing program. Interface actions are those actions specific to particular user interface styles, but can be used in various application domains. Examples are such as scrolling a window or selecting a command button action in a graphical interface style, or typing a character string action in a text-oriented interface style. Mappings from an application action to interface actions define the interface for that particular action. Cartoonist, upon accepting a request on how to perform an "application" task, depends on these mappings to construct the animation scene at the level detailed enough for the current interface style. More detail of the mapping syntax is documented in [Sukaviriya91a].

---

```

Action: connect
Parameters:
  object1 Class object1 Component†,
           NotClass object1 Wire,
           IsPartOf object1 CurrentDesign;
  object2 Class object2 Component,
           NotClass object2 Wire,
           IsPartOf object2 CurrentDesign,
           NotSame object1 object2;
  outLine Class outLine OutputLine,
           IsPartOf outLine object1;
  inLine  Class inLine InputLine,
           IsPartOf inLine object2;

Pre-conditions:
  exist (object1)
  exist (object2)
  hasExceedOutputFanout (object1)
  hasUnconnectedInputLines (object2)

Post-conditions:
  reduceFanout (object1)
  reducedConnectedInputLines (object2)
  connected (object1)
  connected (object2)

```

---

Figure 5 *Connect* Action Representation

With the User Interface Design Environment's knowledge representation scheme [Foley91] as its predecessor, each action in Cartoonist is defined with parameters, pre-conditions – conditions which must be true prior to

<sup>†</sup> Component is a superclass of the class Gate.

performing the action, and post-conditions – consequences of the action. Each parameter is defined with constraints which state the criteria of parameter values acceptable for the action for which it is defined. Figure 5 shows the representation of a *connect* action in the digital circuit layout application. (Parameter constraints of a parameter are listed next to the parameter.) It is these parameter constraints which Cartoonist uses to automatically select appropriate values for action parameters to be used in animations. The selection process varies from randomly selecting from an acceptable range (of an enumerated type or an integer range, for example) to selecting from the current context a value or an object which satisfies the constraints.

The third category of Cartoonist's knowledge is interaction technique. Each interaction technique captures the lexical steps, both input steps and output feedback, of an interaction; an example is shown in Figure 6. In the same fashion as application actions are mapped to interface actions, interface actions are mapped to interaction techniques. By traversing the mappings from application actions to interface actions to interaction techniques, complete lexical detail of how to perform a task is obtained by Cartoonist. While extracting appropriate objects from the current context for an animation scene using parameter constraints as guidance, it is this lexical detail in interaction technique representation which guides the actual animation of displaying symbols such as a mouse with a button down, a keyboard with a key pressed, etc. Descriptions of the architecture of Cartoonist and how these knowledge entities are semantically linked and used to create help scenarios are elaborated in [Sukaviriya90, Sukaviriya91a].

---

```

Technique: MouseClickObject
Parameters: object
Steps:  pressMouseButton (point)
        verifyPointWithinAnyObject (point, object)
        releaseMouseButton (object)
        verifyPointWithinAnyObject (point, object)

```

---

Figure 6 *MouseClickObject* Interaction Technique

### Architecture

In Cartoonist, its help component is tightly integrated with the user interface control component; communication channels have been established which allow contextual information to be accessible by both the help and the user interface components. To maintain consistency of system feedback during each animation, Cartoonist, instead of simulates system feedback as it goes, only animates the input part of each interaction delegating the burden of demonstrating the system feedback to the user interface component. We have designed the knowledge representation model such that a sufficient amount of information is available to animated help to simulate input tokens. Taking advantage of input token specifications (represented with interaction techniques), Cartoonist simulates input tokens, which substitute for the real tokens which would have been generated by the actual input

activities being animated, while animating each input step. The architecture is designed such that the control is switched back and forth between the interface component which generates feedback, and the animated help component for incremental animation.

### Animation

In Figure 1, Cartoonist simply retrieves the definition of the *createNAND* action and randomly selects a location within the current design for the NAND gate to be created. In Figure 2, Cartoonist selects among the existing gates in the design to demonstrate how to rotate, randomly selects among the choices for degrees of rotation, and interacts with the dialogue box object to determine whether there is an OK button. In this case there is, so it animates selecting the OK button to complete the rotation procedure. At the end of each animated help session, Cartoonist prompts the user whether the context before the animation should be restored. If the user does not want to restore the context, Cartoonist takes no actions and easily ends the help session. The context is restored otherwise.

Notice in Figure 5 the pre-conditions in the *connect* action representation, which state that there must be at least two gates for connection, *gate1* must not exceed its fan-out limit, and *gate2* must have at least an input line available for connection. These conditions are satisfied in the context of Figure 3, therefore, the animation of connect can be performed within the context. Had the OR gate selected for the destination for connection in this scenario no available input lines, Cartoonist would select some other gate instead, either the NAND gate or the XOR gate in this case.

When at least one of the pre-condition of an action to be animated is not true, Cartoonist invokes its planner to come up with a plan which satisfies it. The planning algorithm implemented in Cartoonist partially follows the planning model in [Wilkins88]. The general strategy for Cartoonist's planning is to satisfy each unsatisfied pre-condition separately until all conditions are satisfied. At the end, the whole plan is linearized into a sequential list of actions and is passed back to Cartoonist to be animated. Had there be only one or zero gate in the design of Figure 3, Cartoonist would invoke the planner to come up with actions which would make two gates exist in the design.

---

Action: selectObject  
Parameters:  
    object Class object ApplicationObject;  
Pre-conditions:  
    visible (object)  
Post-conditions:  
    highlighted (object)

---

Figure 7 *SelectObject* Action Representation

Figure 4 demonstrates the case where all pre-conditions of the *connect* action has all its pre-conditions satisfied, yet the context is still not appropriate for connection, specifically at the interface level. In this case, the

*selectObject* action (shown in Figure 7), which is bound to the task of both selecting an input line and output line of a gate for *connect*, requires that the gate be visible for selection. The planner comes up with a *scroll-window* action which could bring back the gate to the visible range. Cartoonist has to compute how far to scroll the window based on the location of the invisible gate – locations of objects are part of the context which it shares with the user interface component.

### RELATED WORK

LeFevre's study supported the fact that people would use examples, if provided, first and sometimes alone, to find solutions to their procedural problems [LeFevre86]. The examples she used in her experiments were pictorial and were similar to her problem domain, thus similarity of examples to problem solving procedures was one of the reasons she concluded why her subjects preferred examples. Booher's experiment proved that subjects receiving graphical instructions, with or without text, were superior in speed [Booher75]. Palmiter's experiments showed that subject trained with animated demonstrations, with or without spoken text, were superior both in speed and accuracy in performing tasks [Palmiter91]. Wickens' compatibility theory, concluded with support from his experiments, indicated high compatibility among visual stimuli, spatial information processing, and manual responses (moving a joystick, for example). These human factor studies support the use of graphics for procedural instructions, both in static and dynamic formats. The older static form was dominated by studies for printed instructions and older, non-graphical computer technology; interactive graphical and multimedia interfaces would require more than static graphics to ensure the compatibility between procedural help and the actual procedures. All these evidences strongly supports animation as one form of help medium. Context-sensitivity is our conjecture for more effective help presentation medium, its utility is undergoing a study as proposed in [Sukaviriya91b].

A number of animated help systems or the like predated Cartoonist. CADHELP used pre-stored animation for a CAD application [Cullingford82]; the animation part for the same system was later improved and implemented as a new animation engine called GAK [Neiman82]. GAK used the same representations used by the explanation system in CADHELP to render the animation; no runtime information was incorporated into GAK's generation process. APEX intelligently rendered graphical illustrations from its knowledge base and displayed graphical symbols to emphasize the instructions on which objects to operate on and in which directions [Feiner85]. APEX's graphical illustrations were a step close to animation. Its focus on automatic generation is similar to that of Cartoonist.

Apple Macintosh's introductory guided tour [Apple88] was a good example of how effective animation could be used. However, the Macintosh animation was pre-stored and the guided-tour package ran as a single application; it was inaccessible when users were actually performing tasks. An

implemented example of "context-sensitive" animated help was introduced in 1988 [Sukaviriya88] using a visual directory tree application as an example. The animation was hard-coded, but served as an inspiration for Cartoonist.

Tuck implemented a guided-task help system using Mickey's user interface representations to reason about interface objects involved in a procedure [Tuck90]. His help system did not use animation, but was location-sensitive in that it directly pointed to graphical interface objects on the screen while providing pre-stored procedural textual instructions. COMET, on the other hand, automatically generated multimedia instructions for equipment maintenance and repair [Feiner90]. COMET dealt with non-computer, physical-object domain – the domain which requires graphics in procedural instructions as much as the graphical interface domain. The latest animated help was reported in [Baecker91] where each Macintosh HyperCard's drawing tool icon contained a mini-animation of how the tool was used; the animation was not context-sensitive and was only displayed within the icon.

### IMPLICATIONS TO MULTIMEDIA HELP

Though human factors studies reported previously seem to strongly support the use of graphics and animation, graphics is not all which is needed to learn computer procedures. Palmiter reported difficulties in task transfer and retention among subjects trained with animated demonstrations, while those trained with text performed better in both transfer and retention. The strength of each medium must be combined to improve user learning. The user interface research community still does not have a full understanding of how to complement between media, both in terms of their information contents, timing, and appropriate selection of media.

Our conjecture is that, while animation vividly provides graphical details at the interface level which is directly useful to mechanically performing the task, the application-related semantics of these animations is what is important to imprinting the underlying concepts learned. Since lexical details are too verbose to say and require users to translate into the visual space of a graphical interface, it is better to use animation to present them. Wickens' compatibility theory supports this conjecture. The un-presentable concepts directly related to animation should then be presented textually or verbally. Competition among human information processing resources and Wickens' theory suggest using multiple media which do not simultaneously compete for the same input channel, the same type of information processing, or the same response mode. This should be incorporated into multimedia presentation strategies. This would mean appropriate selection of media based on a careful analysis of help information contents, how compatible they are to the media used, which media are already in use, how directly related they are to the actual tasks to be performed, and whether the media used are compatible with the nature of the tasks.

### CONCLUSIONS AND FUTURE WORK

We have discussed: the definition of context-sensitive animated help, the system which can automatically generate this kind of help, previous research related to animated help, and its significance in the context of multimedia help.

In addition to integrating animation into a multimedia help environment, our more specific future direction is to use the same mechanism in Cartoonist to generate a short, context-sensitive tutorial session. Based on parameter constraints, the system can provide some general information about the parameter types and then ask users to participate in selecting values for (or selecting objects as) the parameters. If the values are not acceptable, the semantic of constraints can be used as a basis to explain to users why they are not acceptable. By using pre- and post-conditions of actions, the system can search for actions which would satisfy the unsatisfied pre-conditions and ask users to choose among them. This way, when help is requested on an action, of which some pre-conditions are not satisfied, the users would have more control of how these conditions should be satisfied assisted by the system searching for the candidate actions. By being able to participate in selecting different parameter values, help can be used as a way to experiment with "what will happen if I do this?" questions. This is applicable both at the action level, and at the task level with several actions involved.

Another interesting direction is to use context-sensitive animated help as a way to teach users more difficult, situation-specific semantics of an action. When learning a new action, users normally would not want to know every detail of the action. Context-sensitive animation may be confusing at this point for novices. However, specific contexts may require knowing more detail of the action semantics and how to use the action appropriately; once the users have a general idea about an action, context-sensitive animation, with explanation of the rationale of the system selection of certain parameter values for example, would serve well for the users to incrementally learn about the action.

### ACKNOWLEDGEMENT

I would like to thank my advisor, Jim Foley, and all my dissertation committee members for giving me advice and interesting ideas related to this work. I would like to thank the graphics and user interface research group at the George Washington University, where most of this research was done, for the intellectual environment they provided. This research is funded by the National Science Foundation Grant IRI-88-13179, Sun Microsystems, Inc., Siemens Corporation, and the Software Productivity Consortium.

### REFERENCES

- [Apple88] Apple Computer, Inc. A Diskette. *Apple Tour of Macintosh II Applications Disk*. Cupertino, California: Apple Computer, Inc, 1988.

- [Baecker91] Baecker, R., I. Small, and R. Mander. "Bringing Icons to Life." In *Proceedings of Human Factors in Computing Systems, CHI'91*. May 1991, 1-6.
- [Booher75] Booher, H.R. "Relative Comprehensibility of Pictorial and Printed Words in Proceduralized Instructions." *Human Factors* 17,3 (1975): 266-277.
- [Feiner85] Feiner, Steve. "APEX: An Experiment in the Automated Creation of Pictorial Explanations." *IEEE Transactions on Computer Graphics and Applications* 5 (November 1985): 29-37.
- [Feiner90] Feiner, S.K. and K.R. McKeown. "Coordinating Text and Graphics in Explanation Generation," *Proceedings of the 8th National Conference on Artificial Intelligence AAAI'90*, pp. 442-449, 1990.
- [Foley91] Foley, J.D., W.C. Kim, S. Kovacevic, and K. Murray. "UIDE—An Intelligent User Interface Design Environment." In *Architectures for Intelligent Interfaces: Elements and Prototypes*. Eds. J. Sullivan and S. Tyler, Reading, MA: Addison-Wesley, 1991.
- [LeFevre86] LeFevre, J., and P. Dixon. "Do Written Instructions Need Examples?" *Cognition and Instruction* 3,1 (1986): 1-30.
- [Neiman82] Neiman, D. "Graphical Animation from Knowledge." In *Proceedings of AAAI'82*. 1982, 373-376.
- [Palmiter91] Palmiter, S., and J. Elkerton. "An Evaluation of Animated Demonstrations for Learning Computer-based Tasks." In *Proceedings of Human Factors in Computing Systems, CHI'91*. May 1991, 257-263.
- [Sukaviriya88] Sukaviriya, P. "Dynamic Construction of Animated Help from Application Context." In *Proceedings of the ACM SIGGRAPH User Interface Software Symposium*. October 1988, 190-203.
- [Sukaviriya90] Sukaviriya, P., and J.D. Foley. "Coupling a UI Framework with Automatic Generation of Context-Sensitive Animated Help." In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*. October 1990, 152-166.
- [Sukaviriya91a] Sukaviriya, P. *Automatic Generation of Context-Sensitive Animated Help*. A Ph.D. Dissertation. Washington, D.C. : George Washington University, 1991.
- [Sukaviriya91b] Sukaviriya, P. *Multimedia Help: A Literature Survey and a Preliminary Experimental Design*. A Technical Report GIT-GVU-91-18. Atlanta, GA : Georgia Institute of Technology, 1991.
- [Tuck90] Tuck, R., and D. Olsen. "Help by Guided Tasks: Utilizing UIMS Knowledge." In *Proceedings of Human Factors in Computing Systems, CHI'90*. April 1990, 71-78.
- [Wickens83] Wickens, C.; Sandry, D.; and Vidulich, M. "Compatibility and Resource Competition between Modalities of Input, Central Processing, and Output" *Human Factors*, 25(2), pp. 227-248, 1983.
- [Wilkins88] Wilkins, D.E. *Practical Planning: Extending The Classical AI Planning Paradigm*. California: Morgan Kaufmann Publishers, Inc., 1988.