Sketching Storyboards to Illustrate Interface Behaviors

James A. Landay and Brad A. Myers

HCI Institute, School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA
E-mail: landay@cs.cmu.edu
Web Page: http://www.cs.cmu.edu/~landay

ABSTRACT

Current user interface construction tools make it difficult for a user interface designer to illustrate the behavior of an interface. These tools focus on specifying widgets and manipulating details such as colors. They can show what the interface will look like, but make it hard to show what it will do. For these reasons, designers prefer to sketch early interface ideas on paper. We have developed a tool called SILK that allows designers to quickly sketch an interface electronically. Unlike paper sketches, this electronic sketch is *interactive*. The designer can illustrate behaviors by sketching *storyboards*, which specify how the screen should change in response to user actions.

Keywords

Gestures, design, sketching, interaction techniques, SILK.

INTRODUCTION

When designers first start thinking about a visual interface, they often sketch rough pictures of the screen layouts. These screens are often tied together by storyboarding techniques: the designer annotates the sketches to illustrate sequences of system responses to end-user actions. The simple storyboard in Figure 1 illustrates that the rectangle in the window should be rotated when the button is pressed.

Sequencing between screens by using sketched storyboards is a powerful tool for making early concept sketches [1]. In fact, all but one of the 16 designers we surveyed [2] claim to use sketches or storyboards during the early stages of interface design. Storyboards are a natural representation and they can be used to simulate functionality without worrying about how to implement it. The success of HyperCard has demonstrated that a significant amount of behavior can be constructed by sequencing screens upon button presses.

We have developed an electronic sketching tool called SILK which allows designers to illustrate behaviors while the interfaces are still in their rough early stages. We have added a powerful storyboarding mechanism to the basic widget sketching interface that we reported on last year [2]. SILK preserves the important properties of pencil and paper: a rough drawing can be produced *very quickly* and the medium is *very flexible*.

The main advantage of SILK over paper sketches is that it allows the storyboards to come alive and permits end-users to exercise the interface in this early, sketchy state. Buttons and other widgets were active in our previous system (*i.e.*, they would give feedback when clicked), but they could not perform any *actions*. Our new storyboarding component allows a wide variety of behaviors to be illustrated by sequencing screens on mouse clicks.

When it comes to supporting interaction, existing tools fall short of the ideal. UI builders, such as Visual Basic, require programming languages to specify any interaction beyond that of the individual widgets. Design tools, such as Director, allow the sequencing of screens, but lack the fluidity of paper-based storyboarding, and for anything but the most simple sequences they require the use of scripting. Requiring interface designers to use programming or scripting languages is unacceptable for our domain. We tried to design a system that allows the rapid illustration of a significant amount of interaction by sketching alone.

Due to the lack of good tools, many designers use “low-fidelity prototypes” [4]. A drawback to using these is the lack of interaction possible between the paper-based mock-up and a user – a designer needs to “play computer” and manipulate sketches in response to a user’s verbal actions. In contrast, our system performs the screen transitions automatically. This allows more realistic testing of rough interface ideas.

SILK STORYBOARDS

The behavior of individual widgets is insufficient to test a working interface. For example, SILK knows how a button operates, but it cannot know what interface action should occur when a user presses the button. Storyboarding allows the specification of this *dynamic* behavior between widgets and the basic behavior of new widgets or application-specific objects.

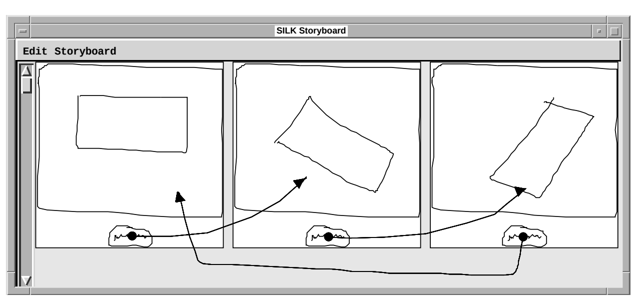

Figure 1: Rotate the rectangle upon button presses.

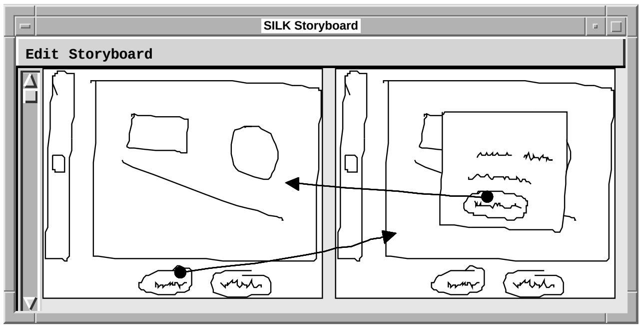

Figure 2: Make a dialog box appear when the button is pressed.

Visual Notation

Our storyboarding technique uses a visual notation that is drawn on and between copies of the interface screens. These sketchy marks are similar to the types of notations that one might make on a whiteboard when designing an interface. Our visual language has two types of objects, *screens* and *arrows*. Each screen is a sketch of an interface in a particular state. Arrows connect objects contained in one screen with a second screen. The arrow indicates that when the object in the first screen is clicked on with the mouse, SILK should display the second screen instead of the first. We believe that this static representation, which can be later viewed and edited, is natural and easy to use, unlike the hidden textual representations used by other systems, such as HyperCard.

For example, Figure 1 illustrates three screens that differ in only the orientation of the rectangle in the drawing window. An important point about this example is that it shows that a designer can illustrate a behavior (*i.e.*, rotation) that the underlying tools, SILK and Garnet, do not even support. Figure 2 illustrates bringing up a dialog box on top of a window. This is interesting since the designer can make the dialog box opaque, thus hiding any objects it appears over. This can also be used for illustrating pull-down menus.

Storyboard Construction and Testing

The designer constructs storyboards by sketching screens with a stylus or a mouse in the sketch window. Screens are then copied to the storyboard. At this point, the original screen can be modified. Now, the designer can start drawing arrows on the storyboard that indicate screen sequencing or more screens can be produced. The free-form arrows can be drawn from any widget, graphical object, or the background to another screen. Thus, the designer can cause transitions to occur when the user clicks on any of these items.

When the designer is ready to test the specified interaction, she can switch to run mode. Now an end-user can start interacting with the sketch and it will make the proper transitions as defined by the storyboard. Each time the user clicks on an object that has the source of an arrow attached to it, the system will replace the current screen with the screen attached to the arrowhead.

In order to allow the designer to debug her storyboards, we have supplied some feedback mechanisms that are displayed while in run mode. First, the currently active screen (*i.e.,* the one displayed in the sketch window), is highlighted in the storyboard window. Second, the object that caused the last transition is highlighted along with the arrow leading to the current screen. A designer can use these mechanisms to help check that her visual program is working properly.

Screen Trees

SILK's storyboarding model implies that a program can be thought of as a tree. The nodes of the tree are the different states of the program (*i.e.*, screens) and the arcs out of each node represent the end-user actions that cause state changes. In order to fully specify a program, the designer would have to specify the entire tree. However, we do not believe this is a major drawback of our model, since storyboarding is used for illustrating *key sequences* in the interface, rather than for specifying an entire interface. For those that require more power, we propose several techniques elsewhere [3] which make it easier to specify more of the screen tree.

STATUS

SILK runs under Common Lisp on both UNIX workstations and the Apple Macintosh. It is implemented using the Garnet toolkit. SILK supports the recognition and operation of several widgets and the transformation of the sketch to an interface with a Motif look-and-feel. The only event the storyboarding system supports is clicking on widgets or graphical objects. Elsewhere [3] we describe how to specify timer events (for animation), double clicking, and other events. We are currently performing user testing of SILK and we plan for design students to use SILK in an interface design course to see how it performs in practice.

CONCLUSIONS

Designers need tools that give them the freedom to sketch rough design ideas quickly, the capability to specify transitions between screens and behavior of interface elements, the ability to test the designs by interacting with them, and the flexibility to fill in the design details as choices are made. SILK was designed with these needs in mind. Unlike paper sketches, our electronic storyboards allow end-users to interact with the sketch before it becomes a finalized interface. SILK storyboarding is a key step to a future in which much of a user interface will be illustrated, specified, and tested by a user interface designer.

REFERENCES

1. Boyarski, D. and Buchanan, R. Computers and communication design: Exploring the rhetoric of HCI. *Interactions 1*, 2 (April 1994), 24–35.
2. Landay, J.A. and Myers, B.A. Interactive sketching for the early stages of user interface design. In *Proceedings of CHI '95: Human Factors in Computing Systems,* Denver, CO, May 1995, pp. 43–50.
3. Landay, J.A. and Myers, B.A., "Just draw it! Programming by sketching storyboards," Carnegie Mellon University, School of Computer Science, Technical Report CMU-CS-95-199, November 1995.
4. Rettig, M. Prototyping for tiny fingers. *Communications of the ACM 37*, 4 (April 1994), 21–27.