Stencils-Based Tutorials: Design and Evaluation

Caitlin Kelleher
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
caitlin+@cs.cmu.edu

Randy Pausch
Entertainment Technology Center
Carnegie Mellon University
Pittsburgh, PA, USA
pausch@cmu.edu

ABSTRACT

Users of traditional tutorials and help systems often have difficulty finding the components described or pictured in the procedural instructions. Users also unintentionally miss steps, and perform actions that the documentation's authors did not intend, moving the application into an unknown state. We introduce Stencils, an interaction technique for presenting tutorials that uses translucent colored stencils containing holes that direct the user's attention to the correct interface component and prevent the user from interacting with other components. Sticky notes on the stencil's surface provide necessary tutorial material in the context of the application. In a user study comparing a Stencils-based and paper-based version of the same tutorial in Alice, a complex software application designed to teach introductory computer programming, we found that users of a Stencils-based tutorial were able complete the tutorial 26% faster, with fewer errors, and less reliance on human assistance. Users of the Stencils-based and paper-based tutorials attained statistically similar levels of learning.

ACM Classification Keywords: H5.2. Information interfaces and presentation (e.g., HCI); Graphical User Interfaces (GUI); Training, help and Documentation.

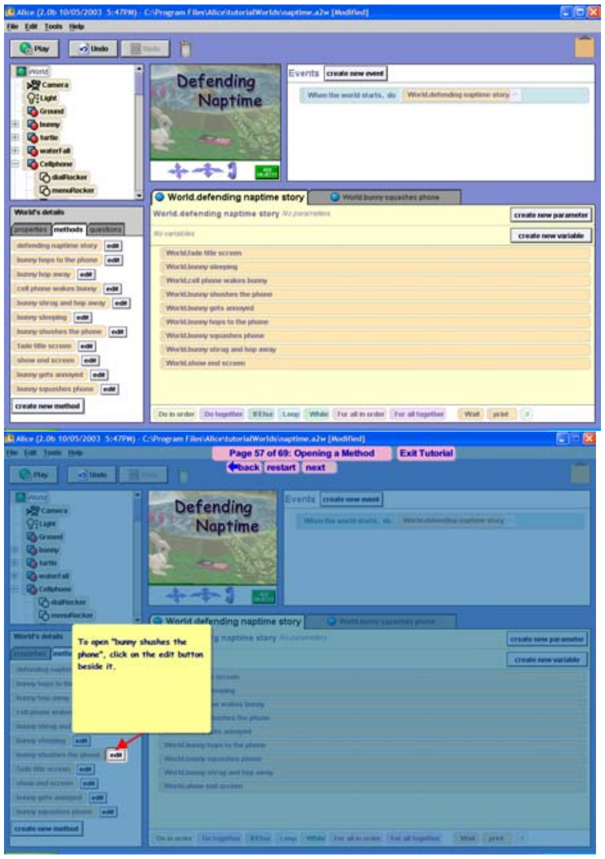
Keywords: Tutorials; interaction technique; transparent overlay; user interface design.

INTRODUCTION AND MOTIVATION

Software applications commonly provide either paper or online tutorials and reference manuals. This documentation contains sequences of written instructions and images that illustrate the steps a user should perform to accomplish specific tasks. Presenting the tutorial in a separate context from the application it is designed to teach creates unnecessary problems for novice users.

Many users have difficulty locating the interface components described or pictured in the instructions [21]. Online documentation creates an additional problem: there are two versions of the component on the screen, real and pictorial. Users will often click on the image of a component in the tutorial rather than on the "real" component in the interface [21].

Further, the list format in which most paper and online instructions are presented creates additional problems for users. Because all instructions for a task are equally visually appealing, users often lose their place in the instructions while switching between the instructions and the application [21]. Based on our user testing, users often inadvertently skip steps and make mistakes.



To open "bunny shushes the phone", click on the edit button beside it.

Figure 1: The Alice user interface without (above) and with (below) a stencil

Skipped steps, mistakes, and unspecified actions can hamper the documentation's ability to teach by moving the application into an unintended state. Because the user can put the application into an unintended state, and writing documentation for all possible states is prohibitively costly, users can find themselves in confusing application states that no longer match the next step of the documentation.

In this paper, we describe Stencils, an interaction technique for presenting tutorials that, in each step, draws the user's eye to the correct screen component and prevents the user from interacting with other components, allowing the user to concentrate on learning the application or feature (see Figure 1). We show the Stencils technique at work in Alice [1], a complex software application designed to teach introductory computer programming.

To evaluate the Stencils technique, we compared the performance of users using a paper-based and a Stencils-based version of the same tutorial. We found that users of the paper-based and Stencils-based tutorials learned the tutorial material equally well. However, users of the Stencils based tutorial completed the tutorial 26% more quickly, made fewer mistakes in completing the tutorial, and were less likely to require human assistance to make progress.

RELATED WORK

There are three relevant areas of related work: the presentation of procedural instructions, learner-centered design, and transparent interfaces. We will discuss work in each of these areas.

Presenting Procedural Instructions

Much of the research on how to present procedural instructions to users has been performed in the context of developing better help systems for software applications. Currently, most applications present procedural instructions for help systems in a separate window with supplementary pictures [15, 16]. However, researchers have found this method to be problematic for users [21]. Users often forget steps while switching between the instruction window and the application, have difficulty locating components pictured in the instruction window, or mistakenly think that the images of interface elements presented in the instruction window are fully functioning components [21]. Since most web-based tutorials use a similar format, it is likely that users of web-based tutorials will encounter similar problems. While users of printed tutorials are unlikely to confuse images of interface elements with the actual interface elements, they may still have difficulty locating the interface elements or accidentally skip steps.

Efforts to improve on-line presentation of procedural instructions have centered on two areas: 1) improve the quality of procedural instructions presented in a separate context, 2) and find ways to present help in context.

Early work on presenting procedural instructions demonstrated that adding pictures to textual instructions helped users complete procedural instructions more quickly, but did not improve their accuracy [6]. Because of the dynamic nature of many user interfaces, researchers have suggested [2, 28] and evaluated [18, 24, 25] using animated demonstrations to present procedural instructions to users. Palmiter et al. found that subjects who used an animated tutorial initially completed test tasks faster than those who used a text-based tutorial, but users of the animated tutorial did not retain their learning a week later [24,25]. Harrison found that users who used animated tutorials or illustrated textual tutorials learned more quickly than users who used a non-illustrated textual tutorial [18]. Researchers have concluded that for many types of software, animated demonstrations will not be broadly effective for presenting procedural instructions [18, 24].

Since many of the problems users encounter when using traditional on-line help or tutorials are caused or exacerbated by the separation between the instructions and the application, other researchers have tried to make help available in the context of the application. Coachmarks [13] are markings, typically a circle, cross or check in red or green, drawn over a component in the interface to attract the user's attention to the component relevant to the current step. Sukaviriya et al. [32] animate the cursor over the interface and replace the typical arrow cursor with representations of the mouse and keyboard to indicate user actions. Coach/2 used an animated picture of a mouse that left a graphical trail and blinked its eyes to show mouse clicks [29]. Both techniques show the user what interface components to focus on. However, users may not fully understand what actions are necessary to accomplish a given task. We have not found any studies comparing the performance of subjects using in-context instructions with that of subjects using more traditional instructions.

While the purpose of procedural instructions is to teach users new skills, once a user has located the relevant set of procedural instructions in a help system, the system may have enough information to perform the instructions for the user. Current versions of the Windows™ Operating System [34] include a "Show Me" feature that automatically performs the steps described in the instruction window without showing the user how the steps were performed. Although this type of feature does not help users learn new functionality, it does give users an option if they are unable to understand and perform the steps described.

Rather than trying to improve the presentation of procedural instructions, some researchers have tried to limit the number and kinds of mistakes that users can make. Carroll and Carrithers [7] found that users using a specially-created training version of a word-processing package learned to use the program more quickly and performed better on a post-test that measured comprehension than users using the unmodified version of the word-processor. In the training system, when users choose an advanced feature in the training version, the system responds with a dialog box stating that the chosen command is not available in the training system. In a later study, Catrambone and Carroll demonstrated that subjects who learned to use the Training Wheels version of the word-processor with the help of a guided-exploration training card were able to transfer their knowledge to an unmodified version of the word-processor [10]. Further, these subjects were able to perform similar and more advanced tasks as quickly as or more quickly than users who learned to use the unmodified version of the word-processor with the same guided-exploration training card [10]. While limiting the number and kinds of mistakes users can make may help them learn new software more effectively, creating and maintaining separate training versions of software is extremely labor intensive since modifications made to the full program must also be made to the training version of that program.

Learner-Centered Design

Researchers in Learner-Centered software are exploring ways to create software-based scaffolding, support for learners as they are learning a new task [30]. While software-realized scaffolding can take many forms, some Learner-Centered systems provide scaffolding that is intended to guide learners through a process such as creating a simulation or researching a question. Emile, a system for building physics simulations, implements process control by enabling menu items that allow users to access parts of the interface relevant for later stages in simulation building only after they have completed earlier stages [17]. TheoryBuilder, a tool for constructing scientific models, uses reminder messages displayed in pop-up windows to remind learners to perform parts of the process they have neglected. Users can request that TheoryBuilder stop reminding them to complete a given task by clicking a “Stop reminding me” button displayed underneath the reminder message [20]. Other systems use the user interface to suggest the process learners should follow but do not require learners to follow it [26,33].

Transparency in User Interfaces

Previous work has examined the use of transparency in interfaces and interaction techniques to solve a variety of user interface problems.

To make better use of screen real estate, Bartlett created stipple-based transparent controls that could exist in an application’s work area without obscuring it [3]. Kramer proposed the use of translucent, arbitrarily shaped regions as an alternative to the overlapping windows paradigm that could more fluidly support design activities [22].

The Stencils technique is most closely related to the work done by Bier et al on the See-Through Interface: both use a transparent layer drawn over a user interface to change how an application responds to interface events such as mouse clicks [4,5]. A See-Through Interface consists of Toolglass widgets and Magic Lens filters that appear as though they are on a sheet of transparent glass in between the mouse cursor and the user interface [4,5]. A Magic Lens changes the appearance of the user interface beneath it by applying a filter, such as *magnification* to it [4,5]. By moving a Toolglass widget over a user interface object and clicking on it, a user can apply that widget’s operation to the selected object [4,5]. By using their non-dominant hands to position sheets containing one or more Toolglass widgets and Magic Lens filters over the user interface and their dominant hands to control the mouse cursor, users can select and operate on interface objects in fewer steps and with less cursor motion [4,5].

Researchers have explored the use of Magic Lenses and Toolglass widgets in several domains including 3D virtual worlds [35], augmented reality [23], generating database queries [14], and debugging user interfaces [19].

APPROACH

Stencils is an interaction technique designed to present tutorial instructions in the application context while preventing many kinds of errors. Stencils-based tutorials present users with sequences of full-screen, colored, transparent overlays (or stencils) containing holes. These stencils appear visually overlaid upon the active application interface and intercept mouse and keyboard events. Events occurring over a hole in the stencil are passed to the GUI component beneath the hole. This prevents users from interacting with components covered by the stencil. The holes in the stencil draw the user’s eye to the component they should interact with during a given step. Notes on top of the stencil can supply additional information.

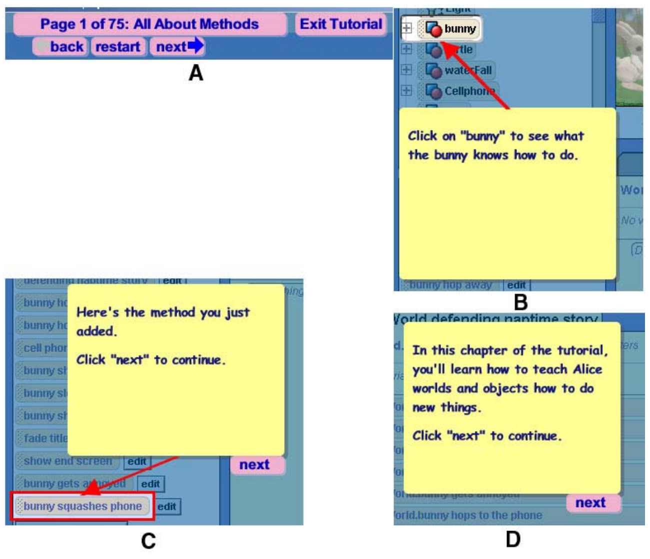
We have created four types of stencil objects for use in creating tutorials or help instructions.

Navigation bars are automatically added to every stencil. They provide “next” and “previous” buttons. The navigation bar also indicates which step the user is currently performing and displays the total number of steps in the current task (see Figure 2A). An “Exit Tutorial” button allows users to close the tutorial at any point.

Holes with attached notes are the most common interface elements. They provide a hole through which the user can interact with the underlying application component and an associated note that the tutorial author can use to provide necessary information. We draw a red arrow to connect the note with its associated hole (see Figure 2B).

Frames with attached notes highlight a particular application component without allowing the user to interact with it. They are typically used to bring aspects of the interface to the user's attention. For example, a frame could point out the results of a completed step. An attached note provides any necessary explanation (see Figure 2C).

Stand-alone notes are used to provide a motivation or describe a goal that will take more than a single step. They are associated only with the stencil, not with any particular element in the application interface (see Figure 2D).



Click on "bunny" to see what the bunny knows how to do.

Here's the method you just added.

Click "next" to continue.

In this chapter of the tutorial, you'll learn how to teach Alice worlds and objects how to do new things.

Click "next" to continue.

Figure 2: Stencil Objects A) Navigation bar B) Hole with note C) Frame with note D) Stand-alone with note

One possible problem with tutorial instruction within the application is that users may confuse interface components belonging to the tutorial with those that are part of the application. To prevent this, interface elements associated with the tutorial have a different visual appearance than standard GUI elements, always appear on top of the stencils, and are slightly transparent so the user can see components in the underlying interface. Based on our user testing, users do not have difficulty differentiating which interface elements belong to the help and which ones belong to the application interface.

Interaction Description

In each step of the tutorial, the interface is covered by a stencil. Directions for the current step are displayed on sticky notes. To aid readability, these notes are almost opaque and are placed by the tutorial author over parts of the interface that are least relevant to the current step. However, notes are movable and the user can reposition them to get a better view of a part of the underlying interface, if desired. The stencil also contains holes over any elements of the interface that the user needs to interact with. Users can perform all necessary actions through the hole. While the rest of the interface is visible, it is not accessible: if users click on elements of the interface that are covered by the stencil, nothing will happen.

Steps in the tutorial are presented one at a time. Users move to the next step in one of two ways: for steps that require a simple action such as a mouse click or an enter key, the stencil will automatically advance to the next step when it detects the user has performed the correct action; for more complex steps, the user presses a "next" button to advance when s/he has completed the step. When users move to the next step in the tutorial, Stencils checks the current state of the application against a saved "correct state" to verify that the user has performed the step correctly. If the user has made any mistakes, stencils displays both a note stating that it believes the user has made a mistake and a "back" button that returns the user to the beginning of the previous step so that they can try again. If the user has correctly performed the step, the system advances to the next step in the stencils-based tutorial.

Occasionally, users want to return to a previous step. To allow this, we provide a "previous" button as part of the navigation bar. When a user returns to a previous step, Stencils takes them to the beginning of that step by undoing all of the actions they have performed as part of the current and last steps. To move forward, users must complete the steps as directed by the tutorial. By undoing changes when the user goes back a step, Stencils ensures that the state of the program is always consistent with the tutorial instructions for that step.

Implementation Issues

Stencils in Alice

Our implementation of Stencils is written using the Java Swing framework. It uses the *glassPane* component in *JRootPane* to draw the stencil over the existing interface and intercept all mouse events. Each stencil maintains a list of holes and components associated with those holes. If a mouse event occurs inside a hole, the stencil passes the event to the interface element below; otherwise the stencil processes the event. Keyboard events are also controlled by explicitly managing which interface elements in the underlying application have keyboard focus. Keyboard events reach an element in the application interface only if that interface element is associated with a hole that has the stencil's focus. A focus listener for the stencil's focused object prevents the user from moving to another interface element using the keyboard.

Modifications to Alice

To enable Stencils-based tutorials in Alice [1], chosen because of its open source status and fairly complex graphical interface, we had to modify the Alice system to implement the Stencils Application Interface, a Java interface that provides system-specific functionality to the tutorial. This functionality includes the abilities to:

1. Request the position and size of an interface element given the name of the element.
2. Request the name of the interface element at a particular position on the screen.
3. Request a string representation of the changes.
4. Ask whether or not two strings representing changes in the world are equivalent.
5. Undo changes made to the Alice world and the interface.

If the layout of the components in Alice changes, Alice alerts the tutorial by calling methods in the Stencils Update Interface (a second Java interface), allowing the tutorial to determine whether any holes or frames in the current stencil need to shift.

Our implementation of Stencils is written in Java and can be used by any Java application (implementations for other languages are possible). It includes a basic authoring tool and the ability to play back Stencils-based tutorials. To use Stencils, a Java application must implement the Stencils Application Interface and make appropriate calls to the Stencils Update Interface to alert the Stencils system to changes in the layout of the user interface.

Authoring Stencils-based Tutorials

We have created a simple authoring tool for building help stencils to allow non-programmers to create Stencils-based tutorials. The authoring tool runs on top of the active application. Objects are added to the stencil by double clicking on its surface. By default, this creates a hole with an attached note. A right click menu allows authors to create a frame with a note or a stand-alone note rather than a hole with a note. The author can reposition notes by dragging them on the surface of the stencil and add information by typing. Notes are visually attached to their associated holes or frames with a line that updates when they are moved.

After creating the necessary holes in a stencil, the author of a Stencils-based tutorial must perform the actions necessary to complete the current step. Stencils then requests a Java String representation of the changes the tutorial author has made during that step from Alice. These change strings are saved for every step in the tutorial such that the tutorial can check users' work as they complete steps.

Advantages

The Stencils technique has several advantages over previous work in the presentation of procedural instructions. Stencils greatly decrease the number and types of mistakes that a user can make. The visual representation of the stencil draws the user's eye to the component for the current step. Each stencil provides a visual indication of what the user can do in that step, without altering the appearance of the application below. The Stencils technique handles complex interactions: pop-up menus appear on top of the stencil and interface components can be dragged from one hole to another. Instructions for each step are superimposed on the interface and displayed a single step at a time. Consequently, users cannot lose their place in the instructions or inadvertently skip steps.

LESSONS FROM FORMATIVE EVALUATION

To gain an understanding of whether or not Stencils was helpful in real applications, we chose to develop and test Stencils in the context of a complex piece of software. Alice is a programming environment that allows novice programmers to create animated 3D virtual worlds by dragging and dropping command tiles [11,12]. Although it has been designed for and tested with novice users, Alice is a relatively complex piece of software with more than 150 clickable interface elements and more than 300 drag-able elements. While not all of these elements are visible at the same time and many are placed in inconspicuous locations in the interface, the Alice interface can still be overwhelming for beginning users.

While developing the Stencils interaction technique, we conducted formative evaluations of three versions of a Stencils-based tutorial with 15 users (7 female), ranging in age from 18 to 60. Users were asked to work through short tutorial segments while talking aloud. The tutorial segments included navigating through the interface, selecting menu options, creating new interface elements, and dragging and dropping interface elements. The primary lessons we learned were:

1. Visually reinforce the stencil as an overlay on top of the interface

We found that it was important to make holes and notes appear slightly 3-dimensional. Without a hint of 3-dimensionality, users sometimes concluded that the interface was simply tinted blue. With a shadow drawn at the holes to indicate depth and under the notes so they visually float above the stencil, users seemed to understand that the stencil was a layer on top of the existing interface.

2. Bring changes that occur underneath the stencil to users' attention.

Simple actions, such as changes in selection, sometimes cause changes in parts of the interface that are underneath the stencil. Because the notes and stencils direct users' attention to particular regions of the interface, users are less likely to notice changes in other parts of the interface. If a particular step directs users to perform an action that will cause a visual change in an area of the interface not exposed by a hole, the next step in the tutorial should use a frame to highlight that change.

3. When completing simple actions, such as mouse clicks or single keystrokes, users expect the tutorial to automatically advance.

We observed that while users seem to prefer to control the pacing of complex actions, they expect the tutorial to automatically advance to the next step when they perform simple actions, particularly mouse clicks. Surprisingly, our evaluations indicated that users were not confused by the tutorial sometimes auto-advancing and sometimes requiring manual advancement.

4. It is necessary to do at least minimal checks to ensure that users have done the right thing.

While the Stencils technique prevents many kinds of errors, it is still possible for users to make errors. Since future steps may rely on elements that are created in past steps, it is crucial to verify that elements used in future steps are created and not deleted.

5. The underlying application needs to alert the tutorial to changes in the layout of the interface.

Some actions the user takes may cause elements in the interface to shift. If any of these elements have holes or frames over them, these shifts may result in holes or frames over the incorrect parts of the interface.

6. For sequences of steps that have holes over the same screen components, shifting the location of the notes provides a cue that users have moved to the next step.

For many users, the change in position of the notes from one step to the next is a cue that they have advanced to the next step. When one step asks the user to manipulate the same interface elements as the previous step, and the notes do not change location, users may conclude that the tutorial did not advance and inadvertently skip a step.

METHOD

We implemented the lessons learned during formative evaluation, and conducted a study to compare the performance of users given Stencils-based and paper-based versions of the same tutorial.

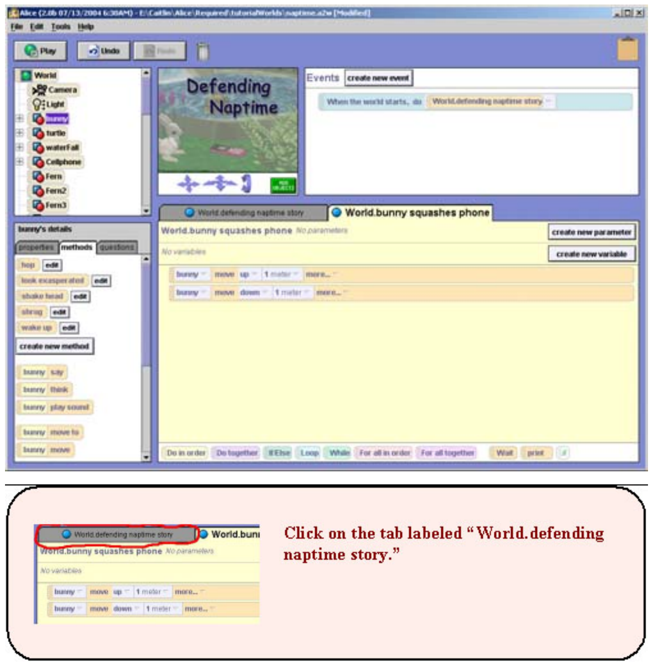
Participants

Twenty-two Cadette Girl Scouts representing three troops from the Pittsburgh area participated in our study. The girls ranged in age from 12 to 16 years, with 18 of the 22 being between 12 and 13. When asked to rate their skill with computers, 5 chose "very good", 14 girls chose "good", 2 chose "fair", and 2 chose "poor or nonexistent". Of the 22 girls, one had prior programming experience, and 7 had experience creating webpages. The study was conducted during three one-day, four-hour workshops (one for each troop). Participants were paid for their participation.

We chose to evaluate the stencils-based tutorial with girls because this study is a part of a larger project to create a programming system that gives middle school aged girls a positive introduction to computer programming. We believe that if we can make the tutorial work for middle school girls, who tend to have less computer experience and less confidence in their computer skills than boys of the same age [30], it will work for many other groups of users. Additionally, we included several infrequent computer users between 40 and 60 in our formative evaluations to ensure that this technique also works for older novice users.

Preparation of Experimental Materials

The paper and Stencils-based tutorials guide users through a sequence of changes to three Alice worlds. The textual directions to users are the same in both conditions.

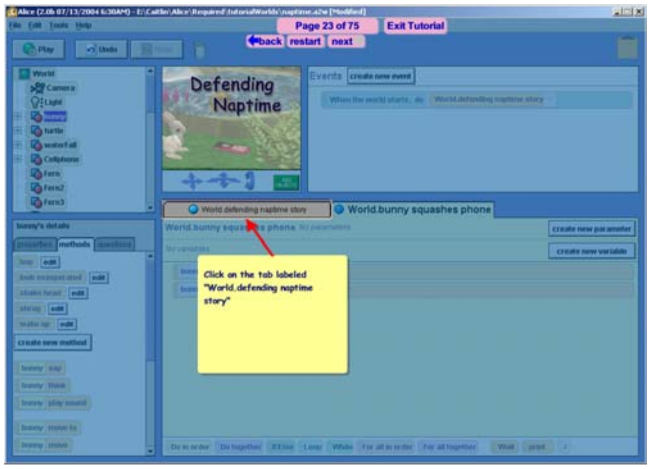


Click on the tab labeled "World.defending naptime story."

Figure 3: Paper tutorial instructions (below) with corresponding Alice interface (above)

Paper-based Tutorial

In the paper version of the tutorial, directions are presented beside a picture of the GUI component the user needs to interact with for that step. Because users often have difficulty locating components on screen, the pictures of each component include enough screen context to allow users to easily identify which of the five regions of the Alice interface, their target component lies in (see Figure 3). To allow users to check whether or not they have correctly completed the steps in the tutorial, we have included images that show what the relevant parts of the Alice interface should look like at several points throughout the tutorial



Click on the tab labeled "World.defending naptime story"

Figure 4: Tutorial step illustrated in Figure 3 as seen by a user of the Stencils-based tutorial.

Stencils-based Tutorial

In the stencils-based version of the tutorial, directions are presented on yellow Post-it™ style notes on the surface of the stencil. Holes in the surface of the stencils draw users' attention to components they need to interact with during the current step of the tutorial. Since our early user testing showed that users often do not notice interface changes that happen below the stencil, the tutorial uses frames to draw users' attention to changes that they have made. When users press the *next* button or the stencil auto-advances to the next step, it checks to make sure that the user has performed all actions necessary for the current step and has not performed extraneous actions.

Procedure

The study took place during three four-hour Alice workshops and used a two-group between-subjects design. Participants were randomly assigned to use either the paper or stencils-based tutorial. To minimize the effects of differences in computer experience or academic potential among the three troops, an equal number of participants from each troop were assigned to the paper-based and stencils-based tutorial conditions. Both the paper-based and stencils-based conditions consisted of 3 participants from troop 1, 3 participants from troop 2, and 5 participants from troop 3, for a total of 11 participants in each condition.

During the workshop, participants completed three tasks: the tutorial, a post-tutorial survey, and a quiz designed to measure tutorial learning. The quiz required participants to perform tasks taught in the tutorial in order to answer multiple-choice questions about an Alice world they had never seen. Participants needed to perform a variety of actions including: playing the world, finding and calling methods, navigating through the gallery of 3D objects supplied with Alice, adding 3D objects to their worlds, and editing predefined methods.

There were no time limits for completing the tutorial, post-tutorial survey and quiz. Participants were instructed not to help each other, but were told that they could ask the experimenter for help with the tutorial, if necessary. The experimenter provided help only when requested.

Data Collection

To enable us to study users' performance on both the tutorial and the quiz, we recorded users' actions in two ways. We instrumented the Alice program to record any changes that users made to the current Alice world. To record actions users took that did not result in changes to the current Alice world, we used a locally developed logging program that saves screen captures and records all mouse and keyboard events. We used the screen shots and event logs to reconstruct videos of the users' computer screens as they completed the tutorial and quiz.

Using both the Alice logs and the videos of users' computer screens, we produced transcripts of all actions the users took while completing the tutorial and quiz. In addition, we recorded the amount of time spent on each tutorial and the quiz.

Dependent Measures

To compare the success of participants using the stencils-based and paper-based versions of the tutorial, we use error rate, elapsed time, and number of requests for help. To evaluate learning, we use the number of correct answers on the quiz and the elapsed time in completing the quiz.

Tutorial Errors

Because we are primarily interested in mistakes that could make users unable to progress, we counted three types of errors: skipped steps, incorrect selections that caused changes to which elements are displayed in the user interface, and incorrect actions that caused changes to the Alice world. Any actions not described in the tutorial that caused a change to either the interface or the Alice world were counted as errors. However, if a user started an action and canceled it without making a change to the interface or the world, that action was not counted as an error. Additionally, if a user made an error but immediately corrected it (e.g. choosing the wrong item from a menu and immediately changing it to the correct one), it was also not counted as an error.

Elapsed Time

The elapsed times for the tutorial and quiz were measured beginning when the user opened the file for a given tutorial and ending when they began to load the next file (e.g. clicked on the File menu) or closed the Alice program.

Results

We used unpaired t-tests to compare the performance of participants using the stencils and paper-based tutorials.

Tutorial Performance

We found that users of the stencils-based tutorial made fewer errors and took 26% less time than users of the paper- based tutorial. Users of the stencils-based tutorial skipped fewer steps ($p = 0.012$), made fewer erroneous changes to the Alice worlds presented in the tutorial ($p = 0.023$) and to the user interface ($p = 0.069$). In addition to making fewer mistakes, users of the stencils tutorial were 26% faster in completing the tutorial ($p = 0.057$): the mean time for completion of the stencils tutorial was 47 minutes, 22 seconds; the mean time for completion of the paper-based tutorial was 59 minutes, 22 seconds. Users of the stencils-based tutorial also were less likely to require human assistance to make progress on the tutorial ($p = 0.08$). The average number of errors and the distribution of error counts are shown in Table 1.

Table 1: Average number of errors and distribution of users' error counts for Paper and Stencils-based test groups.

		Average # Errors per User	# of Users making n Errors					
			0 errors	1-2 errors	3-4 errors	5-6 errors	6-10 errors	>10 errors
Paper	skipped steps	3.82	0 users	3 users	5 users	1 user	2 users	0 users
	interface errors	4.55	5	1	1	2	0	2
	world errors	5.09	1	5	4	0	1	0
	help requests	0.727	7	3	1	0	0	0
Stencils	skipped steps	1.27	5 users	3 users	2 users	1 users	0 users	0 users
	interface errors	1.36	4	5	2	0	0	0
	world errors	1	6	4	1	0	0	0
	help requests	0.08	10	1	0	0	0	0

Quiz Performance

There was no significant difference between the performance of users of the stencils-based and paper-based tutorials on a post-tutorial quiz. Users of the paper-based tutorial answered an average of 5.00 out of 6 questions correctly and users of the stencils-based tutorial answered an average of 4.82 correctly (p = .746).

There was also no significant difference in the amount of time necessary for the users of the stencils-based and paper tutorial to complete the post-tutorial quiz. Users of the stencils-based tutorial took an average of 20 minutes, 17 seconds to complete the quiz where users of the paper-based tutorial completed the quiz in an average of 18 minutes, 24 seconds (p = .721). These averages are based on the completion times for users who performed all steps in Alice necessary to answer the quiz questions (stencils 8 users, paper 6 users).

Survey Results

In a survey about the tutorial given after users had completed the tutorial but before they had started the quiz, we found that users of the stencils tutorial were more confident that they completed the steps in the tutorial correctly (stencils 4.55, paper 3.64 on a 5 point scale p = 0.029). However, users of the paper tutorial were more confident that they could build a world in Alice after completing the tutorial than the stencils-based users were (stencils 3.55, paper 4.18 on a 5 point scale, p = 0.051).

DISCUSSION

The Stencils technique is a potential alternative for presenting tutorials. Based on our data, it allows users to attain the same level of learning in a substantially shorter period of time, with fewer errors, and less reliance on human intervention to make progress.

One of our initial concerns with the Stencils approach was that users might move through the tutorial quickly and without understanding what they were learning. While the users of the Stencils tutorial did complete the tutorial more quickly, they appear to have done so without sacrificing learning. Both the paper-based and Stencils-based tutorial groups performed similarly in the number of correct answers and the amount of time it took to complete the quiz.

The increased speed of the users of the Stencils-based tutorial is probably due, at least in part, to the fact that Stencils presents the tutorial instructions in the context of the application. While paper-based tutorials require less context-switching than many online-tutorials presented in a separate window, in a given step the users of the paper-based tutorial had to find their place in the paper tutorial, read the directions, find the appropriate components on screen, and determine what the directions wanted them to do. Users of the Stencils-based tutorial needed only to determine what the directions wanted them to do.

While the Stencils technique seems to improve performance on the tutorial, the users of the Stencils-based tutorial had lower confidence in their ability to create their own Alice world after completing the tutorial. This is of concern, particularly for populations of computer users who may have lower confidence levels from the outset, such as middle school girls. One potential explanation for this is that Stencils, while preventing many errors, may give users the impression that they need help to interact with the underlying application. This is an unexpected, subtle effect that requires additional research to understand.

One of the most significant potential advantages of the Stencils technique is that users learning from a Stencils-based tutorial required less human assistance than those using a more traditional paper-based tutorial. Employees in most businesses require at least some training on software. Making software training less reliant on human teachers has huge potential cost savings.

We believe that Stencils will be of greatest benefit in interfaces that are highly spatial and primarily point-and-click with some typing.

CONCLUSION

The Stencils technique suggests a method for displaying tutorial and help instructions in the context of the application, avoiding many of the problems created by having tutorial instructions either on paper or in a separate window. Based on our experiences developing Stencils, we provide guidelines for how to make systems like Stencils work for users. A user study comparing the performance of users given a Stencils-based tutorial with that of users given a paper-based version of the same tutorial demonstrated that users of the Stencils tutorial were faster, made fewer errors, required less help from human teachers, and learned the material covered in the tutorial as well as the users of the paper tutorial.

FUTURE WORK

While our preliminary evaluation of the Stencils technique is encouraging, our study focused on the behavior of a small number of users representing a narrow demographic and using a single software application. To verify that the Stencils technique is generally useful, additional studies should be performed to determine the success of Stencils-based tutorials with diverse user groups and applications.

In addition to suggesting a need for more widespread testing, our experiences user testing Stencils as well as prior work in both Learner-Centered Design and software documentation suggest possible ways to improve the Stencils technique.

In the post-tutorial survey, we found that users of the Stencils-based tutorial were less confident in their ability to create an Alice world than users of the paper-based tutorial. One possible explanation for this result is that the support Stencils provides does not decrease as users become more skillful. In Learner-Centered Design, an important property of scaffolding (educational supports for learners) is that it fades over time. An important direction for the future development of Stencils is the design and evaluation of a version of Stencils that fades support as users gain familiarity with certain tasks. Providing support that fades over time may help users to develop the confidence that they can build Alice worlds without assistance.

Currently, Stencils encourages users to learn the system (by completing the tutorial) before creating their own worlds. Prior research has found that many users are reluctant to devote time exclusively to learning a software system [8,10]. Instead, many want to learn the system as they pursue specific end-goals, such as writing a business letter or computing sales statistics [8]. Users will often scan tutorials and user manuals looking for relevant tasks rather than working through them from beginning to end, as their author intended [8, 27]. To support users learning a system while pursuing their own goals, Carroll et al. created the Minimal Manual, which provided users with instructions for typical goals new users have [8,9]. In the context of a word processor, typical goals might be “typing something” or “printing something” [8,9]. By providing steps for small goals, the Minimal Manual enabled users to learn tasks immediately relevant to their larger goals. Studies found that this approach enabled users to learn a system more quickly than a commercial manual [9,10]. Future versions of Stencils should include support for guiding users through tasks within the context of their current worlds. For example, a user who wants to make a particular character in their world disappear should be able to bring up a tutorial that will walk them through the process of making that particular character in their current world disappear. To enable this kind of interaction, in-context Stencils tutorials will need to allow users to specify a context (e.g. particular characters or objects in their world).

ACKNOWLEDGMENTS

We would like to thank Cliff Forlines for his discussion and ideas on early versions of Stencils, our colleagues in the Squeak Research Group (now at Viewpoints Research Institute) for providing a rich research environment in which to develop the initial concept behind Stencils, and our colleagues in the Stage 3 Research Group at Carnegie Mellon for their ideas and support as we implemented and experimented with stencils. Additionally, we would like to thank our anonymous reviewers for their helpful comments and insights.

REFERENCES

1. Alice. http://www.alice.org
2. Baecker, R., Showing Instead of Telling. *In Proc. Of SIGDOC 2002*, ACM Press (2002), 10-16.
3. Bartlett, J. Transparent Controls for Interactive Graphics. WRL Technical Note TN-30, Digital Equipment Corporation, Palo Alto, CA, July 1992.
4. Bier, E., Stone, M., Pier, K. et al. Toolglass and Magic Lenses: The See-Through Interface. In *Proc Computer Graphics and Interactive Techniques 1993*. ACM Press (1993), 73-80.
5. Bier, E. Stone, M. Fishkin, K. et al. A Taxonomy of See-Through Tools. In *Proc CHI 1994*. ACM Press (1994), 358-364.
6. Booher, H.R., Relative comprehensibility of pictorial information and printed words in proceduralized instructions. *Human Factors 17,* 3 (1975), 266-277.
7. Carroll, J. and Carrithers, C. Training Wheels in a User Interface. *Communications of the ACM 27*, 8 (1984), 800-806
8. Carroll, J. and Rosson, M. The Paradox of the Active User. In J.M. Carroll (Ed.), Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. MIT Press, Cambridge, 1987.
9. Carroll, J., Smith-Kerker, P., Ford, J., and Mazur, S. The Minimal Manual. IBM RC 11637, 1986.

10. Catrambone, R. and Carroll, J. Learning a Word Processing System with Training Wheels and Guided Exploration. In *Proc CHI/GI 1987*, ACM Press (1987), 169-174.

11. Conway, M., Audia, S., Burnette, T., et al. Alice: lessons learned from building a system for novices. In *Proc CHI 2000*, ACM Press (2000), 486-493.

12. Dann, W., Cooper, S. and Pausch, R. *Learning to Program with Alice: Beta Version*. Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2005.

13. Designing Coachmarks. http://www.developer.apple.com/techpubs/mac/AppleGuide/AppleGuide-24.html

14. Fishkin, K. and Stone, M. Enhanced Dynamic Queries via Movable Filters. In *Proc CHI 1995*. ACM Press (1995), 415-420.

15. Goodall, S. Online Help: A Part of Documentation. In *Proc SIGDOC 1992*, ACM Press (1992), 169-174

16. Goodall, S. Online Help in the Real World. In *Proc. SIGDOC 1991*, ACM Press (1991), 21-29.

17. Guzdial, M. Software-Realized Scaffolding to Facilitate Programming for Science Learning. *Interactive Learning Environments 4,* 1 (1995), 1-44.

18. Harrison, S. A Comparison of Still, Animated, or Nonillustrated On-Line Help with Written of Spoken Instructions in a Graphical User Interface. In *Proc CHI 1995*, ACM Press (1995), 82-89.

19. Hudson, S., Rodenstein, R., and Smith, I. Debugging Lenses: A New Class of Transparent Tools for User Interface Debugging. In *Proc UIST 1997*, ACM Press (1997), 179-187.

20. Jackson, J., Krajcik, J. and Soloway, E. The Design of Guided Learner-Adaptable Scaffolding in Interactive Learning Environments. In *Proc CHI 1998*, ACM Press (1998), 187-194.

21. Knabe, K. Apple Guide: A Case Study in User-Aided Design of Online Help. In *Proc CHI 1995*, ACM Press (1995), 286 – 287.

22. Kramer, A. Translucent Patches. In *Proc UIST 1994*, ACM Press (1994), 121-130.

23. Looser, J., Billinghurst, M., and Cockburn, A. Through the Looking Glass: The User of Lenses as an Interface Tool for Augmented Reality Interfaces. In *Proc Computer Graphics and Interactive Techniques*. ACM Press (2004), 204-211.

24. Palmiter, S. and Elkerton, J. An Evaluation of Animated Demonstrations for Learning Computer-based Tasks. In *Proc. CHI 1991*, ACM Press (1991), 257-263.

25. Palmiter, S., Elkerton, J., and Baggett, P. Animated demonstrations vs. written instructions for learning procedural tasks: A preliminary investigation. *International Journal of Man-Machine Studies*, 34 (1991), 687-701.

26. Quintana, C., Eng, J., Carra, A. et al. Symphony: A Case Study in Extending Learner-Centered Design through Process Space Analysis. In *Proc of CHI 1999*, ACM Press (1999), 473-480.

27. Rieman, J. A Field Study of Exploratory Learning Strategies. Transactions on Computer-Human Interaction 3, 3 (1996). 189-218.

28. Schneiderman, B. Direct manipulation: A step beyond programming languages. *IEEE Computer 16*, 8 (1983), 57-69.

29. Selker, T., Barber, R., and Kelley, R. Effective, Selective Presentation of Help Material in a Graphical Environment: Experience with COACH/2, a graphical adaptive help system. IBM Tech Report, 1996.

30. Shashaani, L. Gender-Differences in Computer Experience and its Influence on Computer Attitudes. *Journal of Educational Computing Research 11*, 4 (1994), 347-367.

31. Soloway, E., Guzdial, M., and Hay, K. Learner-Centered Design: The Challenge for HCI in the 21st Century. *Interactions 1*, 2 (1994), 36-48.

32. Sukaviriya, P., Isaacs, E., and Bharat, K. Multimedia Help: A Prototype and an Experiment. *Ext. Abstracts CHI 1992*, ACM Press (1992), 433-434.

33. Wallace, R. Soloway, E., Krajcik, J. et al. ARTEMIS: Learner-Centered Design of an Information Seeking Environment for K-12 Education. In *Proc CHI 1998*, ACM Press (1998), 195-202.

34. Windows Family. http://www.windows.com.

35. Viega, J., Conway, M., Williams, G., and Pausch, R. 3D Magic Lenses. In *Proc UIST 1996*, ACM Press (1996), 51-58.