Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 178 (2007) 171–179
www.elsevier.com/locate/entcs

SSEA: A System for Studying the Effectiveness of Animations

Eileen T. Kraemer[1] Bina Reed Philippa Rhodes
Ashley Hamilton-Taylor

Computer ScienceDepartment
The University of Georgia
Athens, GA USA

Abstract

This paper describes SSEA (System for Studying the Effectiveness of Animations), an environment designed to support the empirical study of program visualizations.

Keywords: Program visualization, Engagement taxonomy, Automatic question generation.

1 Introduction

Many programmers, instructors, and programming students have a strong intuitive belief that visualization is valuable for communicating information about the state and behavior of programs.

In our work we seek to investigate attributes of algorithm animations and other program visualizations that affect how well the user can understand the concepts the designer intends to convey. We believe that effective PV systems must support perceptually appropriate graphical design, layout, and animation, as well as good pedagogical design. We are working to identify and evaluate perceptual, attentional, and cognitive features of program visualizations that affect viewer comprehension. This work is performed in the context of a larger project that involves observational studies of instructors, empirical studies of the perceptual properties of low-level animation actions[1] through the VizEval environment[3], and the development of improved presentation and interaction techniques for program visualization in the context of computer science education.

[1] Email: eileen@cs.uga.edu

doi:10.1016/j.entcs.2007.01.048

In this paper we describe SSEA, a System for Studying the Effectiveness of Animations, an environment designed to support the empirical study of program visualizations. SSEA was created as a testing environment for studying the effects of various attributes of visualization design on viewer comprehension. Through SSEA researchers can select a design characteristic and then create and evaluate a suite of animations exposing variations of that characteristic. Through such experiments the researcher can evaluate the effect of these variations on the viewer's comprehension of the underlying algorithm, as well as the viewer's perception of and attention to selected aspects of the animation. SSEA allows users to view, interact with, and answer questions about an animation, and records the viewer's interactions and responses to questions. Researchers can then examine the log files generated and perform analysis of the responses and timings with respect to the attribute being examined.

2 SSEA System

SSEA integrates a visualization interface with a question panel, pop-up questions and a monitor. The monitor automates collection of data about user interactions and events, producing a log of user interactions and responses.

A SKA (Support Kit for Animation) [4] module manages the display of the animation and adjusts the visualization to any user interactions. SKA is an algorithm animation system designed to support the instructional task and includes graphical primitives and an animation "engine" as well as a data structures library, user interface for the invocation of SKA methods and data structures in an instructional setting, and support for interactions desirable for the instructional task. In the context of SSEA we make use only of the graphics and animation engine, and note that use of the SKA graphical primitives and animation capabilities permits us to directly transfer our results from SSEA studies into refinements of and additions to SKA.

The main SSEA user interface is seen in Figure 1. It includes an animation area (A), a pseudocode display (B), animation controls (C), and a question area (D). A high resolution monitor is required to view the multiple areas of SSEA in their entirety.

The animation area(A) and pseudocode area(B) display the graphical representation of the underlying algorithm, and are synchronized by SKA.

The playback of animations can be controlled by the viewer via the animation control area(C). One feature allows the user to select from a collection of data sets as the input for the algorithm. Another control sets the speed of the animation. The animation can be paused, ended, and then begun again from the start. Stepping through the animation, which can only occur if the animation is paused, causes the next step of the animation to execute before pausing the animation again. A slider indicates the progress of the animation. Moving the slider to the left will allow users to select a point at which to restart the animation.

The experimenter may specify that "popup" questions be displayed while the

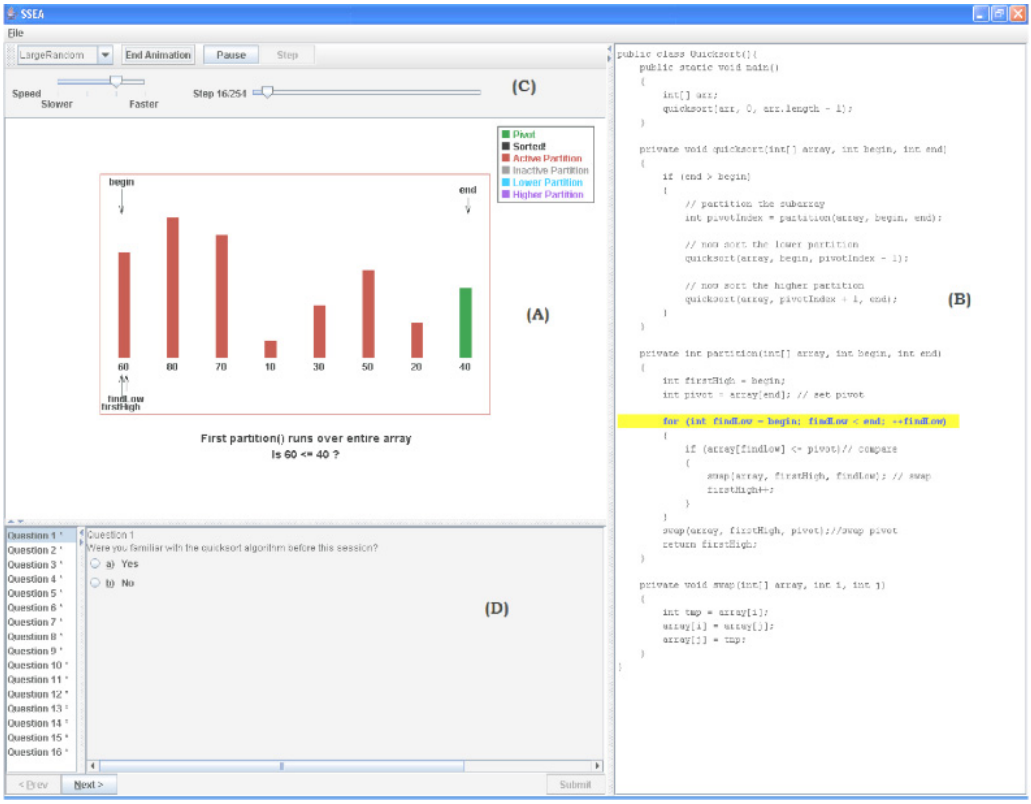

Fig. 1. SSEA screen shot. (A) Animation area where visualization of underlying algorithm is displayed. (B) Pseudo code display: highlights code being executed by underlying algorithm, which is synchronized with animation area. (C) Animation controls: viewer controls playback of algorithm animation. (D) Question listing: viewer responds to questions designed by the researchers to evaluate the viewer's comprehension of the algorithm.

participant is watching the animation. At the time that the popup occurs, the animation pauses its run until an answer is provided. These questions can be associated with the animation of the algorithm on a particular data set. Further, the experimenter may specify that the popup appear only during the initial run of the animation of the algorithm with the associated data set. To prevent users from using the pseduocode to supply the correct answer to the popup questions, the popup window is positioned over the pseudocode area and is not movable. The experimenter also has the option to display the correct answer or other feedback after the participant submits a response. User answers to these popup questions are recorded in the log.

The experimenter may also specify a questionnaire to be displayed initially and may require viewers to answer all questions before proceeding. This feature is typically used to collect demographic or other background information from study participants.

The graphical visualization generated by the SKA module results from an algorithm and an animator working somewhat independently. A threaded architecture is used, following the oroducer-consumer design pattern. An algorithm thread is the producer of data for visualization, while the animator thread consumes the data.

The algorithm thread is created when a new input set is selected. A new animator thread is created for each new run of the visualization.

The graphical representations consist of graphical objects and actions on one or more of these objects. Graphical objects consist of lines, rectangles, text labels, circles and composite graphics. Each object has numerous properties that can include color, fill, visibility, font, position, and labels. The display canvas references a list of graphics. As graphics are updated by the animator module, the canvas repaints the graphics, causing an animation.

The question area (D) contains the questions designed by the experimenter to evaluate the participants' comprehension of the algorithm. These "traditional" questions are displayed individually, with a listing of all questions off to the left side. An asterisk next to a question number in the list indicates that the question is still unanswered. Participants may return to a question and change their response. When all questions are answered, a *submit* button becomes enabled. When the user clicks the *submit* button the session terminates. Both the intermediate choices and the final answers for all questions are recorded in a log.

Details of the implementation of the SSEA architecture and the procedure for specifying an experiment may be found in [2].

3 Findings using SSEA

In this paper we describe an initial study, conducted using SSEA, in which we evaluate the effect of two attributes of an algorithm animation. The first attribute is the presence of cueing (flashing) to indicate to the viewer that two data elements have been compared. We label the conditions as "cueing" (C) and "no-cueing"(X). The "cueing" depiction flashes the compared bars three times when the comparison event occurs. "No-cueing" simply does not flash.

The second attribute is the type of animation used to indicate that two data elements have exchanged values. We label the conditions as "grow"(G) and "move"(M). In the "grow" depiction of an exchange of elements, the bar representing the smaller element grows to the height of the larger element while the larger element simultaneously shrinks to the height of the smaller element. In the "move" depiction of an exchange of values, the two bars representing the data elements being exchanged are seen to move in an arcing motion, each one moving to the location of the other.

Four animations were created: cueing with move(MC), cueing with grow(GC), no cueing with move(MX), and no cueing with grow(GX). An interesting aspect of this study is that we sought to simultaneously evaluate the perceptual effects of these animation techniques and the effects of the animation techniques on comprehension of the algorithm depicted in the animation. By perceptual effect we refer to the viewer's ability to see and understand what is animated (i.e., did users detect that two bars just flashed?, do they understand that the flashing means that the two bars are being compared to one another?, etc.) Perceptual effects are evaluated through responses to popup questions .

We evaluated the impact of these attributes both on perception of the animated changes and on viewer comprehension of the depicted algorithm, as measured by the number of correctly answered questions in two question sets: "traditional" (comprehension) questions and "popup" (perception) questions.

3.1 Method

3.1.1 Participants

Participants in the experiment were 59 volunteers drawn from the undergraduate computer science students registered at the University of Georgia in the fall semester of 2005 and spring semester of 2006. Participants received a five-dollar payment in return for their participation, which required approximately one hour.

3.1.2 Apparatus

Studies were conducted on Dell Dimension desktop computers with high-resolution 17-inch LCD flat-panel color monitors. Experiments were conducted with the SSEA environment.

3.1.3 Stimuli

Participants viewed a quicksort animation, as seen in Figure 1. The elements of the array of values are represented as filled rectangular bars. The value of the array element is indicated by both its height and a label that appears below the bar. The height of each bar (in pixels) is the value of the element multiplied by 22 pixels, a height difference which was determined through pilot studies to be detectable at a confidence level of 95 percent.

The quicksort algorithm is recursive and involves dividing and sub-dividing the array into partitions. An outlining rectangle indicates the current partition. In addition, labeled arrows marked "begin" and "end" indicate the boundaries of the current partition. Two additional labeled arrows marked "firstHigh" and "findLow" represent the variables that move within the current partition looking for values that are higher or lower than the pivot.

Bars are colored to indicate the state of the algorithm. A legend is provided in the upper right-hand corner. The current pivot is colored green. Inactive partitions are colored gray. The active partition is initially colored red. As the algorithm proceeds the bars are sorted into a lower partition (values less than or equal to the pivot) which is colored light blue, and a higher partition (values greater than the pivot) which is colored purple. The pivot value is then swapped with the rightmost value in the lower partition, which is its final location. It is then colored black to indicate that it is in its sorted position.

Two lines of text below the array serve as captions, providing a description of the current step being performed. Lines in the pseudocode display of section B are highlighted as the animation is executing the corresponding event.

3.1.4 Design

Two between-subject factors were varied: cueing and exchange animation. Participants were randomly assigned to one of four groups, each with its own corresponding animation. Fourteen participants were in the cueing with move(MC) category, sixteen in cueing with grow(GC), twelve in no cueing with move(MX), and seventeen in no cueing with grow(GX).

3.1.5 Procedure

An instruction sheet led each participant step-by-step through the study. A sheet explaining the animation controls of SSEA was attached. Each participant initially completed a training exercise in which they used the SSEA system to view a simple "Find Max" algorithm. This allowed the participants to learn how to interact with the animations using SSEA. This training exercise included the use of two data sets, and required that the participants answer one pop-up question and four traditional questions. The instructions led each participant through the use of each of the animation controls.

After completion of the training exercise the participant was instructed to start the quicksort SSEA program. Participants were unaware that others were viewing different versions of the animation. The first step in the main exercise involved completing a questionnaire about the student's gender, year in school, and the computer science classes they had completed or were currently taking.

Participants then viewed the animation for the group to which they had been randomly assigned. Participants in each group were asked the same comprehension-based questions. All questions were multiple choice and reflected a range of concepts related to comprehension of the quicksort algorithm. The instructions directed participants to use the "LargeRandom" input set. This ensured that participants would be asked all eight popup questions, which were categorized as either "cue-specific" (asking which two bars where just compared) or "exchange-specific" (asking which two bars just exchanged values).

Upon completing the traditional questions and selecting *submit*, participants were given the opportunity to comment on the animations they viewed, the SSEA system, or give any general feedback, through a paper survey form. After feedback forms were collected, students received payment for their participation.

3.2 Results

3.2.1 Popup Questions: Perception and Attention

Overall performance on the pop-up questions was 66.22%. Table 1 shows the per-animation group scores for the pop-up questions.

An ANOVA analysis was performed. Cueing was found to have a statistically significant effect $F(1,57)=4.44$, $p < 0.04$ on participant performance on popup questions. Further, cueing was also found to have a statistically significant effect in a subset of the popup questions classified as cueing-specific (1,2,3,8), $F(1,57)=10.39$, $p< 0.002$.

Group	Average Score(%)	Average Time(min)
MC	72.86	26.00
GX	58.41	21.24
MX	62.33	25.25
GC	71.63	20.25

Table 1
Per-animation-group averages for score on the popup questions, and average time to complete experiment.

Exchange type was not found to have a statistically significant effect on performance on the overall set of popup questions. However, in a subset of questions classified as exchange-specific (4-7), a statistically significant benefit to "move" over "grow" was found. $F(1,57)=5.74$, $p < 0.02$.

3.2.2 Traditional Questions: Comprehension

Participants overall performed reasonably well on the traditional questions, some of which required detailed knowledge of the procedure or time complexity analysis, with an average score of 60.74 percent. Average time to complete the study was 22.9 minutes.

Group	Average Score(%)	Average Time(min)
MC	61.57	26.00
GX	60.41	21.24
MX	59.08	25.25
GC	61.63	20.25

Table 2
Per-animation-group averages for score and time.

An ANOVA analysis was performed. No statistically significant effects were found for performance on overall traditional questions for cueing, exchange type, or interaction effects.

The traditional questions were further sub-divided into groups based on the type of knowledge the question tested, which we labeled Knowledge, Comprehension, and Application. ANOVA analyses were performed on these subsets. Again, no significant difference was found among the four animation groups.

3.2.3 Discussion

The lack of a significant effect on viewer comprehension for the type of flash cueing evaluated in this study may, at first, seem a surprising result. However, a viewing

of the animation (or a review of the animation description provided here) reveals that the animation employs several types of cueing to indicate that two bars are being compared. In particular, color, labeled arrows, and location within the current partition all also cue the identity of the bars to be compared. Thus, we do not conclude that such flash cueing is not valuable in promoting comprehension of animations. Rather, we conclude only that the use of flash cueing as a redundant cue in this animation did not significantly benefit comprehension.

It is interesting to note that flash cueing was found to have a statistically significant benefit both in overall performance on the popup questions and especially on performance on the cueing-specific subset of the questions. The popup questions focused on perception and recall of animation events that had just occurred. The cueing-specific questions took the form of "Which two bars were just compared?". Thus, we can conclude that flash cueing does help the viewer to better note the low-level behavior of the animation. Whether this low-level benefit carries over to the higher-level comprehension of the depicted algorithm appears to depend on the presence of other, redundant cues. In the presence of multiple other cues, as in our study, the flash cueing was not shown to be of significant benefit to comprehension of the algorithm. In the case that flash cueing were the only cue that the values of two bars are about to be exchanged, we speculate that such cueing would quite likely have a much greater impact.

Again in a similar fashion to cueing, the type of animation used to depict an exchange was not found to have an effect on performance on the traditional questions. The lack of a significant effect of exchange type on viewer comprehension of the depicted algorithm, while less surprising, may have a similar explanation to that of cueing: the exchange of bars is cued redundantly. Color, labeled arrows, and position within the current partition all serve to indicate the identity of the bars that have been exchanged. Exchange type did not have a significant effect on overall performance on the popup questions. However, on the exchange-specific subset of the popup questions a significant benefit was found for the "move" animation versus the "grow" animation. These questions took the form, " Which two bars were just exchanged?". We controlled the time aspect of this portion of the animation to ensure that the "grow" and "move" animations required the same amount of time, and can thus eliminate differences in time-on-screen as a possible explanation for this effect.

Perhaps a more likely explanation is that while the bars remain in place in the "grow" animation, they move across the screen in the "move" animation. If the user's gaze has fixated on a portion of the display that does not contain the bars that are exchanging values, then the "move" animation has the potential to move the bars across the user's current area of focus, while the "grow" motion does not. Further, the greater overall motion associated with the "move" animation has the ability to attract the user's attention, even in the periphery of the user's view [Bartram01]. Further studies in which the distance from the user's current focus and between the exchanged bars are varied could be performed to help sort out the components of this effect.

An analysis of the correlation between between performance on the popup questions and on the traditional questions was performed and found to be moderate. This moderate correlation between performance on the popup questions and performance on the traditional questions suggests that the presence of such popup questions may help to focus the user's attention on the details of the algorithm, and help the user to form a better understanding of the depicted algorithm. Another possible explanation for the correlation is that test subjects who perform well on one type of question are "good students" who are likely to perform well on other types of questions. We are investigating this question in a between-subjects study, in which we compare groups who are presented with popup questions against those who do not see popup questions. In addition, we study the effects of the presence/absence of feedback (providing the correct answer to the popup after the user has submitted their answer) and the effect of "predictive" questions (What is about to happen?) versus "reactive" questions (What just happened?). In this work we must also consider the possibility that over-attention to low-level detail may prevent the user from "stepping back" to gain a higher-level, conceptual view of the algorithm's behavior.

4 Conclusions and Future Work

The SSEA environment provides good support for carrying out empirical studies of program visualizations. Several studies have been conducted to data, one of which is reported here. Support for sound actions has been added to the SKA package and additional studies that look at the roles of voice-over and non-speech audio in animations are under development.

References

[1] Davis, E. T., K. Hailston, E. Kraemer, A. Hamilton-Taylor, P. Rhodes, C. Papadimitriou and B.-A. Garcia, *Examining perceptual processing of program visualization displays to enhance effectiveness*, in: *Annual Meeting of the Human Factors and Ergonomics Society*, 2006, in submission.

[2] Reed, B., "Investigating Characteristics of Effective Progam Visualizations: A Testing Environment and the Effect of Comparison Cueing and Exchange Techniques On Viewer Comprehension in Algorithm Animations," Ph.D. thesis, The University of Georgia (2006).

[3] Rhodes, P., E. Kraemer, A. Hamilton-Taylor, S. Thomas, M. Ross, E. T. Davis, K. Hailston and K. Main, *Vizeval: An experimental system for the study of program visualization quality*, in: *Visual Languages and Human-Centric Computing*, 2006, in submission.

[4] Taylor, A. H. and E. Kraemer, *Ska: Supporting algorithm and data structure discussion*, ACM's 33rd SIGCSE Technical Symposium on Computer Science Education **34** (2002), pp. 58–63.