

AN EXPERIMENT DESIGNER TOOL FOR EVALUATION OF PROGRAM
VISUALIZATION QUALITY

by

SUJITH THOMAS

(Under the Direction of Eileen Kraemer)

ABSTRACT

Program Visualization refers to the graphical representation of a program in execution. They are used to facilitate better understanding of the underlying algorithm or other program behavior. However, doubts exist about the usefulness of program visualizations and the extent to which the visualizations are employed in practice [6]. Some studies of program and algorithm visualizations have shown that visualizations have helped the instructional effort [17, 6], while others argue that they have little or no effect on the learning of subjects [19, 21]. Explaining the discrepancy is part of the research to be carried out. One theory is that the quality of the program visualization used has varied among experiments. An experiment generation application called TestCreator has been developed to design experiments that assist in identification of the attributes associated with program visualization quality and quantification of the effects of these attributes on the ability of visualizations to facilitate learning.

INDEX WORDS: Program Visualization, Perceptual Studies, TestCreator, VizEval.

AN EXPERIMENT DESIGNER TOOL FOR EVALUATION OF PROGRAM
VISUALIZATION QUALITY

by

SUJITH THOMAS

B.Tech., University of Madras, India, 2002.

A Thesis Submitted to the Graduate Faculty of the University of Georgia in Partial Fulfillment of
the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2004

© 2004

Sujith Thomas

All Rights Reserved

AN EXPERIMENT DESIGNER TOOL FOR EVALUATION OF PROGRAM
VISUALIZATION QUALITY

by

SUJITH THOMAS

Major Professor: Eileen Kraemer

Committee: Liming Cai
Dan Everett

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2004

DEDICATION

For Mom, Dad, Rajesh, Chechi and Google!

ACKNOWLEDGEMENTS

This thesis was a collaborative effort in the true sense of the word. While my name appears as the author, several members of the VizEval project assisted in the completion of this project.

Thanks to Dr. Eileen Kraemer for her support and guidance. This thesis would have never been a reality if it weren't for her. Also thanks to Matthew Ross, Ashley Hamilton Taylor, Philippa Rhodes and Bina Reed. They have provided support on every decision I made during the course of the project. Guys you were the best to work with!

Last but not least, thanks to Dr. Elizabeth Davis and Ken Hailston for providing me with valuable feedback on the TestCreator application.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	viii
CHAPTER	
1 INTRODUCTION	1
1.1 MOTIVATION.....	2
1.2 PURPOSE.....	4
2 BACKGROUND AND RELATED WORK	5
2.1 EMPIRICAL STUDIES OF PERCEPTION AND COGNITION.....	5
2.2. TOOLS FOR PERFORMING STUDIES	10
3 VIZEVAL SUITE.....	17
3.1 SKA.....	19
3.2 TEST TAKER	20
3.3 FILE CREATOR.....	22
3.4 TEST CREATOR.....	23
3.5 FILE FORMAT	27
4 EXPERIMENT 1: A PILOT STUDY.....	36
4.1 AIM	36
4.2 EXPERIMENTAL SETUP	36
4.3 DISCUSSION.....	38

5	CONCLUSION AND FUTURE WORK	40
	5.1 CONCLUSION	40
	5.2 FUTURE WORK	40
	BIBLIOGRAPHY	43

LIST OF FIGURES

	Page
Figure 3.1: Architecture of the VizEval experimentation platform.....	18
Figure 3.2: Screenshot of the SKA animation module	19
Figure 3.3: Screenshots of the TestTaker module during experimentation.....	20
Figure 3.4: Screenshot of the TestTaker displaying information.	21
Figure 3.5: Screenshots of the FileCreator module	23
Figure 3.6: Pictorial representation of the components of TestCreator module	24
Figure 3.7: Screenshots of the TestCreator module	24
Figure 3.8: Screenshot of TestCreator, highlighting the “number” feature.....	25
Figure 3.9: Screenshot of the TestCreator module, in batch processing mode.....	26
Figure 3.10: Sample graphic and animation files	28
Figure 3.11: Section of a sample Experiment test file	31
Figure 4.1: Pictorial representation of Cue-Change combinations	37
Figure 4.2: Graphical representation of results.	38

CHAPTER 1

INTRODUCTION

Visualization is an old term which has received a large amount of interest recently in the computer science community. Visualization, in the presentation sense, is not a new phenomenon. It has been used in maps, scientific drawings, and data plots for over a thousand years. Examples of this are a map of China (1137 a. d.) and the famous map of Napoleon's invasion of Russia in 1812, by Jacque Minard [16].

Program visualization has been defined by several authors [39, 40, 41], but the general consensus is that it is the use of animated views of computer programs to enhance the human understanding of the behavior and properties of those programs. These visualizations may depict data structures, abstract properties of algorithms, animated source code, or performance behavior.

There are many reasons for the emphasis on program visualizations. The existence of inexpensive computers with substantial graphic-processing features has fueled the interest in the program visualization applications. But perhaps the most important factor is the general consensus among computer scientists that visualization could indeed be a very powerful tool to provide valuable insight into the design, structure and behavior of a program.

However, results from studies have been mixed and inconclusive [6]. While some studies clearly demonstrate the power of visualization to enhance the learning experience [17, 6, 10, 7], others have concluded that there is little or no difference at all [19, 21, 24].

One possible reason for these conflicting results is the differing quality of program visualizations used in the experiments. This raises questions of what constitutes quality in such visualizations. We define quality loosely to mean those attributes that co-relate with the ability of a visualization to convey a desired concept. In this work, we seek to identify the attributes of program visualizations that contribute to quality, to explore the various values those attributes may take on, and to quantify the effects of those attributes and values on the ability of viewers to perceive and, understand and learn from program visualizations.

1.1 MOTIVATION

An NSF funded study, entitled *Program Visualization: Using Perceptual and Cognitive Concepts to Quantify Quality, Support Instruction and Improve Interactions*, conducted in part at the Kraemer Lab at the University of Georgia [42], seeks to answer many important questions about the process of perceptual tracking, attentional processing and conceptual mapping when viewers watch an animation.

- Are viewers able to clearly perceive objects on the screen (can they distinguish individual objects, determine color, size, pattern of motion, determine number of objects, relationships between moving objects)?
- Are viewers able to map objects on the screen and their properties onto the program entities and their properties?
- Are viewers able to correlate events in the animation with events in the algorithm? What is the effect of speed/pacing?
- Is prior or concurrent guidance (in the form of audio explanations, text, visual cues) helpful in interpreting visual phrases?

- Does the process of cognition affect the pace at which users can profitably view an animation? To what extent?

The study focuses on the principle that an effective program visualization system must support perceptually appropriate animation, graphical design and layout, as well as good pedagogical design. In order to address this problem, the study seeks to evaluate perceptual and attentional aspects of program visualization. The principal goal of the study is to identify many of the attributes that are involved in visualization and to experimentally determine the effects of these attributes on the program visualization's ability assist the viewer in understanding the behavior of the visualized algorithm or program. In order to achieve this goal, the study focuses on

- Investigating the characteristics of perceptual tracking, attentional processing, and conceptual mapping relevant to program visualization.
- Gaining insight into the nature and effects of perceptual limitations relevant to visualization of processes, with particular emphasis on program visualization.
- Gaining insight into the effects of perceptual limitations in the face of varying levels or types of cognitive load.
- Evaluating the efficacy of existing PV and non-PV techniques for use in visualizations.
- Defining quality metrics for PV.
- Developing design guidelines that apply specifically to program visualization.
- Developing taxonomy of types of information that one might try to convey via PV and an associated taxonomy of techniques for conveying that type of information.

- Producing an evaluation matrix of the intersection of information type with visualization technique.
- Proposing how these guidelines might be applied to visualization in general.

1.2 PURPOSE

The focus of this thesis is to develop an application that facilitates easy creation of experiments to be used to conduct perceptual, attentional and conceptual studies. This application is embodied in a Visualization Experimentation platform called **VizEval**. VizEval is designed to provide a comprehensive solution for designing, testing and evaluating experiments involving program visualization. This thesis specifically focuses on a part of the VizEval platform called *TestCreator*. TestCreator is a module of the VizEval platform that facilitates easy design, creation and generation of experiment files.

CHAPTER 2

BACKGROUND AND RELATED WORK

Observation and evaluation of prior studies is an integral part of the larger project of which this work is a component. More specifically, the research focuses on evaluating previous studies with insight from the field of perceptual psychology. The study primarily focuses on empirical studies of perception and cognition and evaluating tools that were used for performing studies of perception, cognition and program visualization.

2.1. EMPIRICAL STUDIES OF PERCEPTION AND COGNITION

Research on the effectiveness of algorithm animations in supporting the instructional effort includes work on several low level attributes such as color [4], cueing and motion [2] and high level attributes such as algorithm animations [6] and multimedia [8]. The following section will attempt to discuss a few of these attributes.

2.1.1. LOW LEVEL ATTRIBUTES

Bartram, in her doctoral thesis [2], studied the use of motion as a medium for visually conveying information. She concluded that motion has the unique ability to attract attention over a large visual field and offers a very rich graphical vocabulary. However, she also found that while most types of motion are useful in attracting attention, certain types of motion can be highly distracting to the subject.

Sears and Pylyshlyn studied the relationship between visual indexing and attentional processing [4]. Their experiments involved cueing and tracking multiple objects. They concluded that the test subjects identified changes more accurately when

the target objects were cued. They also found that the efficiency of the subjects decreased as the number of the objects on screen increased.

2.1.2. HIGH LEVEL ATTRIBUTES

As discussed earlier, many scientists believe that animations and hypermedia can facilitate and enhance the learning experience. However, studies that have been conducted are inconclusive. The following sections will briefly talk about some of the research on using animations and multimedia in education.

2.1.2.1. STUDIES OF ALGORITHM ANIMATIONS

Perhaps the single most important application of program visualization in the field of Computer Science education is in algorithm animations. One fundamental quality of a good software engineer is having a good knowledge on the underlying algorithms. However, a large number of students find the subject of algorithms increasingly complex and difficult to understand. Computer Scientists, mainly guided by intuition, have looked into the possibility of using animations as an effective instructional tool to aid in teaching algorithms.

A study by Hundhausen [24] evaluated twenty four studies which focused on studying the effects of program visualizations on learning. He concluded that 43% of the studies concluded that there were significant improvements in learning when the program visualization was used. 8% of the studies found a significant improvement, but they were unable to ascertain as to which attributes of the study contributed to the learning improvement. 4% of the studies found significant results, but in the wrong direction and 42% of the studies found that there were no significant improvements to learning when program visualizations were used.

An early study by Stasko et al. concluded from post test results that there exists a very small advantage for participants using an XTango visualization and text description of pairing heap algorithms compared to a group of students who had access to only the text of the same algorithm [18].

In 1996, Lawrence, Stasko, et al. conducted a more comprehensive study on the effect of visualizations [6]. They concluded that students who attended a lecture, and then created data sets individually while viewing a related algorithm visualization lab exercise fared significantly better on a post-test than those students who attended only the lecture. Similarly, students who were given a data set (did not create their own), fared marginally better than students who attended only the lecture. However, those students who attended lectures taught through slides fared better than those who attended lectures in which algorithm visualizations were used. This study indicates that algorithm visualizations can be used as a very powerful tool in a post-lecture lab-exercise type scenario to reinforce the subject's knowledge of the algorithm.

Hansen et al. conducted a study using a hypermedia (pseudo-code, text, illustrations & visualization) environment known as HalVis [10]. They concluded that subjects who were taught using this environment fared significantly better on the post-test than the subjects who used a textbook. However, their study remains inconclusive on pointing out the exact aspects of the environment that contributed to the result.

Byrne, et al. conducted a series of experiments to measure the effect of making predictions about the next state of an algorithm with and without the aid of visualization [5]. They concluded that while students who attended lectures taught with the use of

visualizations tend to perform significantly better on smaller, less complex algorithms, there was little or no difference for larger, more complex algorithms.

Kehoe, et al. investigated the use of visualizations and other media as a problem solving resource [7]. They conducted a study in which the students were given a text book extract, pseudo-code, and either algorithm visualizations or a series of static images captured from the algorithm visualizations. No time limits were specified. They concluded that the group of students who used visualizations spent more time comparing and moving between different media types and performed significantly better on the post-test.

Another study by Price, et al. was designed to observe the effectiveness of algorithm animations as a debugging medium [19]. They focused on attributes such as debugging time and whether the bug was identified. They concluded that algorithm animations offered no significant differences in this context.

Crosby and Stelovsky studied the effects of using algorithm visualization and multimedia on a classroom lecture [20]. They found that students performed significantly better on a post-test when they were exposed to lectures containing multimedia.

Gurka conducted a study specifically focusing on the learning medium [21]. He wanted to investigate a portion of the study conducted by Byrne, et al. [5]. He conducted experiments on two groups of students using animations and static images. He found that there were no significant differences between the groups on the post-test. However, he noted that the animation group appeared to be highly motivated by the animation.

Kann et al. conducted a study to investigate the level of learner involvement when subjected to program visualization, algorithm animation and constructing algorithm

animations [22]. They found that participants who viewed animations scored significantly higher on post-test than participants who did not view the animation.

Mulholland conducted a study focusing on tracing an algorithm [23]. He used graphical tracers as well as textual tracers. The study allowed five minutes to trace an algorithm and the number of problems solved by both groups was calculated. He concluded that the group that used graphical tracers fared significantly worse than the group that used textual tracers.

Hundhausen and Douglas conducted a study on the level of learner involvement [24]. They allowed test subjects to self-construct visualizations or to actively view pre-defined visualizations. They observed the accuracy and time on tracing and programming tasks for both groups. They concluded that there was no significant difference between the groups on the post-test.

Jarc, et al. investigated the theory of interactive prediction [25]. They used two animation software packages, one that enables the prediction of the next algorithm step and one that does not enable the prediction of the next step of the algorithm. They concluded that there were no significant differences on the post-test between the two groups. They also mentioned that students who used the prediction-enabled software spent significantly more time using the animation software when compared to the non-prediction group.

2.1.2.2. MULTIMEDIA

Faraday and Sutcliffe investigated the use of multimedia as a tool to support the instructional task [8]. They conducted an experiment in which they presented a multimedia presentation to a group. Eye trackers were attached to each person and the

movement of the eyes was monitored. The results indicated that a significant part of the audience was not focusing on the important parts of the presentation. The researcher's then modified the presentation to include attractor elements that would make the audience focus on the key parts of the presentation. As expected, post-test results showed a significant increase in the number of subjects who were better able to understand the content of the presentation. They concluded that multimedia can be an effective instructional tool provided that the presentation is designed in accordance with certain guidelines that emphasize the content to be presented.

2.2. TOOLS FOR PERFORMING STUDIES

A number of program visualization and algorithm animation tools and utility programs have been developed as a part of research or as commercial applications. They can be broadly classified into two types.

- Tools and Utilities to perform program visualizations and algorithm animations.
- Tools for conducting empirical studies on perception and cognition.

The following sections will briefly discuss some of the tools in these categories.

2.2.1. TOOLS USED FOR ALGORITHM ANIMATIONS.

Polka is a visualization system developed by Stasko, et al. at the Georgia Institute of Technology [26]. It allows the user to view animations of algorithms and programs. One of the key features of Polka is its ability to view animations of algorithms and programs that execute in parallel. Samba is the front-end interpreter for Polka [27]. Samba is designed to execute regular ASCII commands. One command is written per line, and animation is performed after each line is read in. Samba was designed to allow

animation files to be readable by any program, allowing for ease of transition to multiple programming language platforms.

SKA (Support Kit for Animation) was developed by Taylor, et al. at the Georgia Institute of Technology and the University of Georgia [28]. It is designed based on the examination of tasks performed in the process of discussing algorithms and data structures. SKA attempts to support and enhance time consuming instructional tasks such as tracing and data structure diagram manipulation, while requiring minimal preparation and authoring time.

ANIMAL is a data structure and algorithm visualization tool developed by Guido Roessling at the Darmstadt University of Technology [29]. ANIMAL is a compact, efficient and easy to use animation tool implemented in Java. ANIMAL is an acronym for A New Interactive Modeler for Animations in Lectures. The tool boasts the following features.

- Very easy to use graphical interface incorporating *drag and drop*
- Animations can be automatically generated using either the scripting language *ANIMALSCRIPT* or the *ANIMALGENERATOR API*.

Matrix is a tool used for the visualization of data structures and algorithms. It was developed by Korhonen at the Helsinki University of Technology [30]. Matrix is an algorithm simulation framework, which provides an extensive component library that allows the user to manipulate fundamental data types such as arrays, linked lists, trees, graphs, and their composites. The system stores the manipulation process, thus allows the user to create algorithm animations without writing any code.

JAWAA was developed at Duke University by Rodger, et al. [31]. JAWAA is a scripting language for creating algorithm animations on the web. Written in Java, the program provides an interface through which users can write animations and then display them with any web browser that supports Java. The animations are written by users in a simple script language that can easily be learned by people with little or no programming experience. For more advanced users, JAWAA commands can be added to their program to quickly produce an animation of a data structure such as an array, stack, queue, graph or tree.

Many current algorithm animation systems have their roots in BALSAs, developed by Marc Brown et al. in 1984 [32]. Its influence has been enormous, and by looking at its design one can learn about the problems that any algorithm animation system must solve. BALSAs and BALSAs-II are general purpose algorithm animation systems, meaning that in principle they can be used to animate any algorithm. They are primarily used to supplement the teaching of computer programming.

After completing work on BALSAs, Brown et al. developed Zeus, an object-oriented algorithm animation system similar to BALSAs [33]. They added several features to Zeus that make implementation easier. These include Zume, a custom pre-processor that generates procedure definitions for renderers based on a file of prototype events, and which thus makes it easier to annotate an algorithm with event calls. They also implemented a multi-view editor, which simultaneously displays a graphical image of a view and the textual code that specifies it. The programmer can manipulate either the graphics or the text and observe the result. This frees the programmer from the need to

specify precise positions for graphical elements. The user can also change the behavior of the algorithm while it is executing, simply by manipulating one the views.

Developed by Stasko at the Georgia Institute of Technology, XTANGO is a general purpose algorithm animation system that supports programmers developing color, real-time, two dimensional, smooth animations of their own algorithms and programs [34]. The focus of the system is on ease-of-use. XTANGO utilizes the path-transition animation paradigm, which helps move animation design to an abstract, high level. Programmers need not be graphics experts to develop their own animations.

CAT and JCAT were developed by Marc H. Brown and Marc A. Najork [35]. CAT stands for Collaborative Active Textbooks. It was designed to act as a Web-based algorithm animation system for an electronic classroom. CAT augments the expressive power of Web pages for publishing passive multimedia information with a full-fledged interactive algorithm animation system. It improves on previous Web-based algorithm animations by providing a framework that makes it easy to construct new animations, including those that involve multiple views. Because views of the same running algorithm may reside on different machines, CAT is particularly well-suited for electronic classrooms. This strategy is an improvement over the electronic classroom systems, which simply display the same X window on multiple machines. JCAT is a Java-based implementation of CAT animation system.

2.2.2. TOOLS USED FOR CONDUCTING EMPIRICAL STUDIES

These software tools and utilities provide the functionality to implement and execute full scale experiments which can then be used to conduct empirical studies on the effectiveness of the animations or on low level attributes such as perception and

cognition. The software tools in this category fall into two broad categories based on the context in which they are used.

- Tools for conducting low-level empirical studies in visual psychology.
- Tools for evaluating the effectiveness of algorithm animations as an instructional medium.

2.2.2.1 TOOLS FOR CONDUCTING LOW LEVEL EMPIRICAL STUDIES IN VISUAL PSYCHOLOGY.

ViA is a perceptual visualization tool designed and developed at the North Carolina State University by Christopher G. Healey and Robert Amant [11]. ViA is designed to help users construct perceptually optimal visualizations to represent, explore and analyze large, complex, multi-dimensional datasets. The system harnesses the low-level human visual system to support rapid and accurate visualization. The system incorporates a formal taxonomy to enumerate dataset properties and to explore and analyze the tasks to be performed. A given dataset is evaluated on its perceptual salience through the use of an evaluation engine. It also tests the usability of mixed-initiative search algorithms that allow external hinting for a collection of mappings and compares the constructed mappings to those set as standards by the domain experts.

Unlike all of the tools discussed above, E-PRIME is a commercial tool developed by Psychology Software Tools, Inc [36]. It is designed specifically for the purpose of conducting experiments in the field of Visual Psychology using an automated environment. It features six basic components namely,

- E-Studio – used to design and develop experiments
- E-Basic – the scripting language for the application suite

- E-Run – the execution environment
- E-Merge – allows merging of different sessions
- E-Data Aid – provides core functionality critical to security and transformation of data
- E-Recovery – used for data recovery from a partially terminated experiment.

The system also boasts a millisecond timing accuracy. However, E-Prime implements its run-time environment in E-Basic, a language that is specific to E-Prime. This limits its usability in terms of support for multiple platforms.

2.2.2.2. TOOLS FOR EVALUATING THE EFFECTIVENESS OF ALGORITHM ANIMATIONS AS AN INSTRUCTIONAL MEDIUM

HalVis is an animation embedded hypermedia visualization tool designed and developed at the Auburn University by Narayanan, et al. [37]. The creators of HalVis believe that, for algorithm animations to be effective, they have to be chunked into meaningful bite-sized pieces and embedded within a context and knowledge providing hypermedia (structure containing other media such as text, diagrams and audio besides annotations) information structure.

HalVis is implemented using the Asymetrix Tool book. It is composed of five tightly integrated modules, namely: fundamentals, conceptual view, detailed view, populated view and questions. The HalVis system provides the ability to embed animations within a hypermedia visualization that also employs textual descriptions, audio narratives and static diagrams to provide contextual information. It also incorporates the functionality to support three distinct kinds of animations to illustrate qualitatively different views of algorithm behavior. The system also features rich

interactions with the animations and uses probes or questions that stimulate thinking or foster self-explanations. On the downside, a substantial amount of time needs to be invested to design hypermedia visualizations, making its use in the classroom very limited. Additionally, there has been no implementation of large, complex algorithms in the HalVis system indicating its low scalability.

CHAPTER 3

VIZEVAL SUITE

The VizEval suite is developed as a collaborative effort between the Department of Computer Science, University of Georgia, and The School of Psychology, Georgia Institute of Technology [1]. The application is designed to provide an experimental platform for assessing and evaluating program visualizations by empirically evaluating the effects of the attributes of program visualization on viewer comprehension. It allows the experiment architect to design experiments and write the specification of all of the components such as graphics, animations, and questions that will be utilized in the experiment. The application consists of four core modules and several utility plug-ins, they are:

- Support Kit for Animation (SKA).
- File Creator.
- Test Creator.
- Test Taker.
- Utility Package (e.g. log file analyzer tool).

The experimental process involves specification of the experiment, running the experiment on the test participants (subjects), and then analyzing the results. The experiment consists of a number of trials. In each trial, the participant views a short animated graphical display and is then asked a series of questions about what they saw and understood. Figure 3.1, depicts the components of the VizEval suite that support this

experimental process. The graphic objects and their animations are specified in graphic and animation files respectively. The context of the experiment is specified in a test file, and the result of a subject's participation in an experiment is stored in a log file.

Typically, creating an experiment involves using the TestCreator module. Here, the experiment architect specifies the various components of an experiment such as the associated graphic and animation files, questions, etc. For large experiments involving a substantial number of graphics and animation files, the FileCreator module may be used to design the objects, specify their animation sequences and generate the graphic and animation files. The TestTaker module takes in a test file generated by the TestCreator and the associated animation and graphics file. It then executes the experiments using the SKA animation package for displaying the animations and the parameter values and conditions specified in the test file. It also keeps track of user responses, timing and other attributes and writes them down to a log file. The log file is then parsed by utility programs designed to extract and analyze data, compare results, etc. The following sections will talk about the different modules of the VizEval suite.

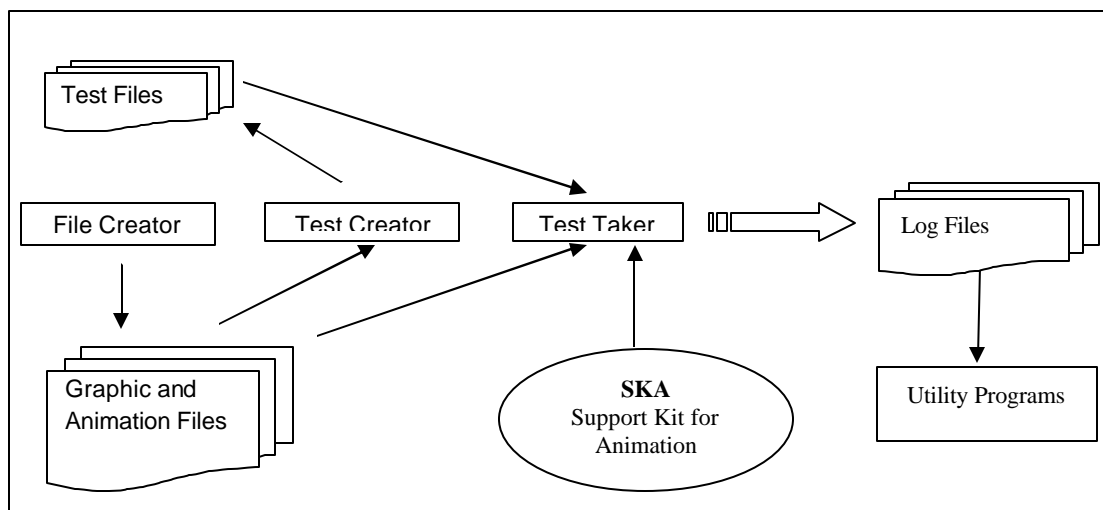


Figure 3.1, Architecture of the VizEval experimentation platform

3.1. SKA

SKA is an acronym for “Software Kit for Animation”. It is the underlying animation and graphics engine of the VizEval suite. SKA consists of a visual data structure library, a visual data structure diagram manipulation environment, and an algorithm animation system and it is prepackaged with a library of a visual data structure classes. Manipulations using SKA includes, adding and removing elements, adding labels to elements, highlighting parts of the visual data structure, and performing other operations specific to a data structure class. SKA also supports duplication; a visual data structure may be duplicated, and the duplicate can be manipulated independently. Each visual data structure instance has a correlated model data structure instance. A visual data structure may be saved and retrieved later.

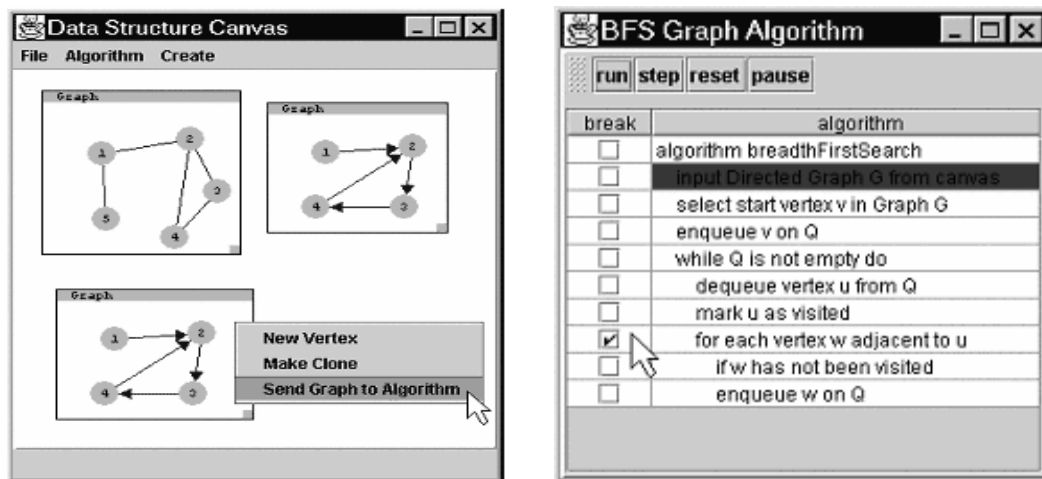


Figure 3.2, Screenshot of the SKA animation module

The environment has the functionality to execute animated algorithms. Through the use of an execution panel, the user can run the algorithm, pause and resume, set and remove breakpoints before or during execution, or step through the algorithm one line at a time. The user can also terminate an algorithm during execution. An algorithm can use,

as input, a visual data structure on the canvas created by the user and it can be manipulated by the algorithm. SKA supports parallel processing; multiple algorithms can be run in parallel, including multiple instances of the same algorithm.

3.2. TEST TAKER

TestTaker is the execution environment for the experiments that were created using TestCreator. The module takes as input, the graphics and animation files that were created using the FileCreator and experiment files that were designed using TestCreator. It logs the user data, date and time at which the experiment was conducted, and other miscellaneous metrics such as height and width of the screen, distance of the eyes from the screen, directory onto which the log files are written into, etc.



Figure 3.3, Screenshots of the TestTaker module during experimentation

TestTaker integrates the data contained in the graphic and animation files and represents it using an internal data structure. It then uses SKA to display the parameters specified in the graphics file and then animates the graphics as specified in the animation file. Once the animation is complete, the trials and questions associated with that specific animation are displayed on the screen. The response of the user is written into a log file along with additional parameters such as date/time, question, correct choice, etc. A useful

feature of TestTaker is its ability to display informative messages to the viewer because it helps in reducing viewer anticipation and anxiety, enabling participants to focus and respond better on the experiment. Some of the types of messages which are supported by TestTaker are

- Pre-test messages are displayed, before the experiment begins .
- Post-test messages can be displayed after the conclusion of the experiment.
- Similarly, messages can be displayed at the beginning and ending of each block or trial. These messages could be used to provide the test subjects with information on their progress such as the number of questions in the upcoming trial, the number of trials left in the experiment, expected time to completion, etc.

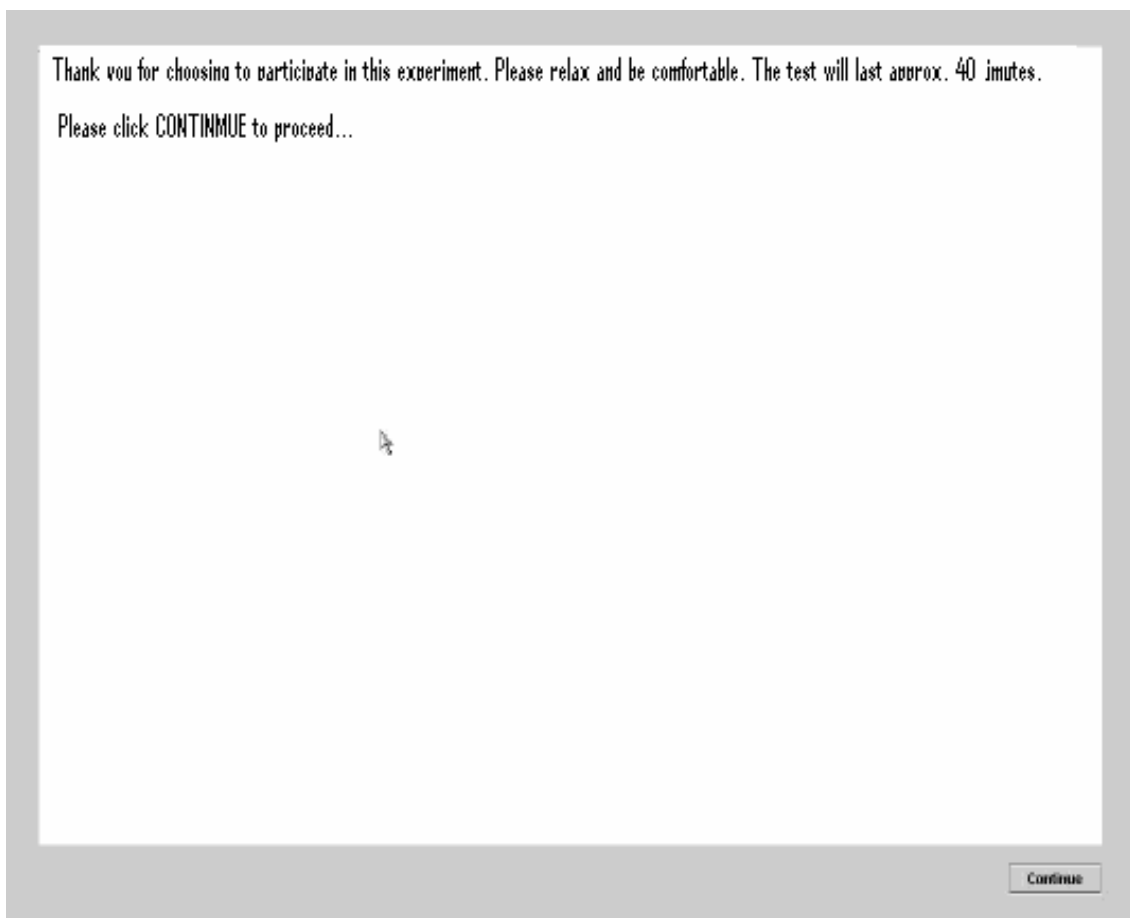


Figure 3.4, Screenshot of the TestTaker displaying information.

TestTaker also features a “data recovery” design in which the responses to the questions are compiled and written to a log file at the end of each trial. In the event of a system failure, data loss is limited to the last trial the test subject was working on. Similarly, the experiment file is executed one trial at a time, facilitating on-the-fly modifications of experiment files. It also features a modular design which is highly customizable. Following this approach, the application can be customized to suit varying testing needs with little or no modification to the code.

3.3 FILE CREATOR

The FileCreator module is used to facilitate easy creation of graphics and animation files that are used in an experiment. In its current implementation, it is designed to rapidly generate graphics and animation files for use specifically in the VizEval project. However, it can be extended to meet other graphics and animation requirements with minimal modification to the code. The user interface of the FileCreator application allows the experimenter to select the number of objects to be displayed on screen. Its value is pre-specified to a set of 4, 8, or 16 objects. The experiment architect can specify the parameters associated with each object such as dimensions, shape, color, and density or allow the software to randomly generate parameter values. In the current implementation, the color, location and step size of the objects are predetermined and the user cannot alter these using FileCreator, but if needed, the graphics file can be opened and manually modified.

The FileCreator can also be used to generate animation files. Animations correspond to objects changing color, height, flashing or moving across the screen, etc. The experiment architect has to specify the type of cue and other parameters specific to

the cue. The application allows the experiment architect to create a single file in which the objects to cue and the objects to animate can be specified or randomly selected.

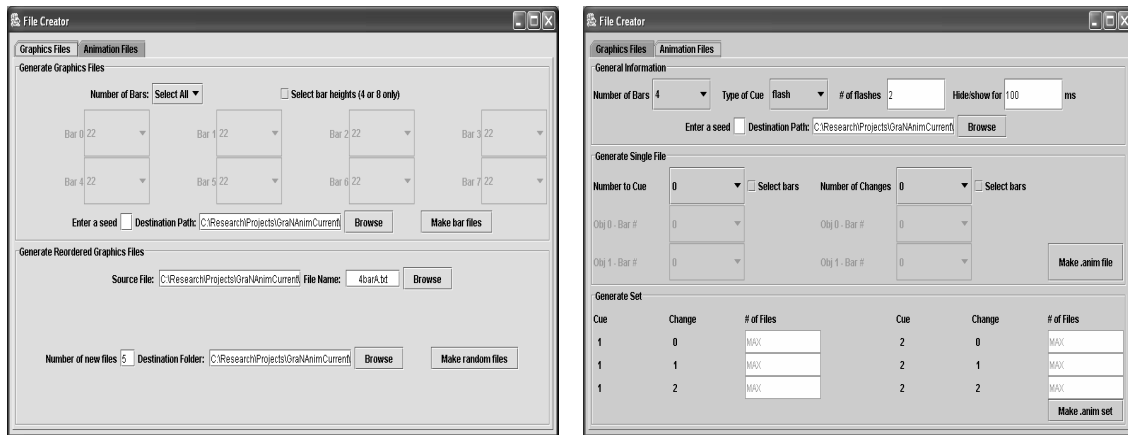


Figure 3.5, Screenshots of the FileCreator module.

When opting to generate a set of files, all possible combinations of animations will be generated for each object. The corresponding files are placed into directories labeled Four, Eight, and Sixteen, respectively.

3.4 TEST CREATOR

TestCreator is a component of VizEval that is the focus of this thesis. The TestCreator application is designed to facilitate easy design and generation of experiment test files. The architecture of the application features a three-tier modular design that supports extensibility and customizability. The application consists of three core modules:

- The user-interface module – contains classes pertaining to the GUI of the application.
- The controller module – contains core classes such as data-structure classes relevant to question types and attributes associated with them, classes associated with the program control, flow & states, etc.

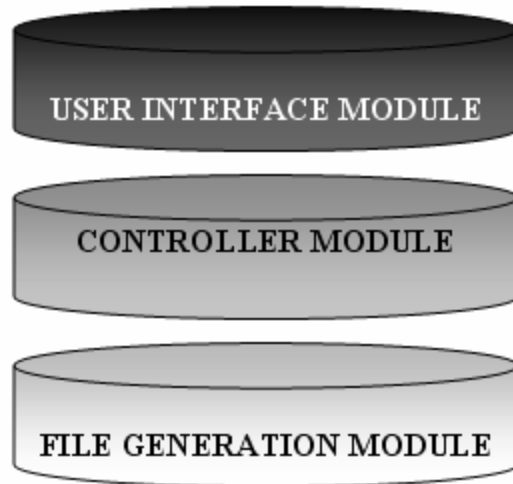


Figure 3.6, Pictorial representation of the components of TestCreator module

- The file-generation module – contains classes relevant to file generation and processing.



Figure 3.7, Screenshots of the TestCreator module.

The GUI features a professional tabbed pane look-and-feel in which the parameters associated with a particular block/ trial or question are kept in the focus of the user, hiding the irrelevant parameters, thus enabling the experiment architect to concentrate on the relevant parameters and at the same time providing the functionality for a quick-lookup of all associated parameters and their values. The design also uses lists that display the questions, trials and blocks of an experiment. Modifying or deleting a block, trial or question is a simple process of picking it from the list and clicking on a button. The application is also designed to generate random values (within a range) for certain parameters, thus saving the experiment architect from having to calculate values for these parameters.

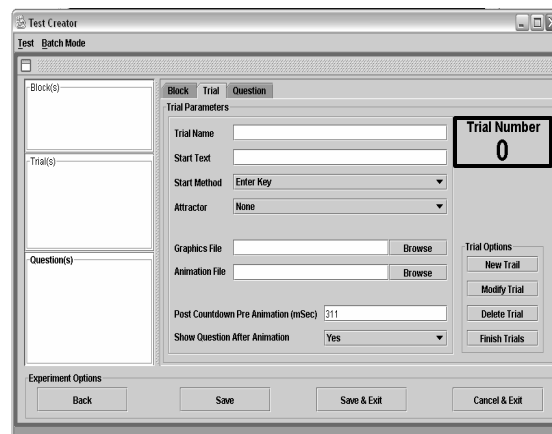


Figure 3.8, Screenshot of TestCreator, highlighting the “number” feature.

Another useful feature is the display of the block, trial or question number within the tabbed panes. This acts as a quick reference for the current block, trial or question that the user is working on, thus avoiding confusion.

The controller module is the nucleus of the TestCreator application. It acts as a bridge between the other two modules. It handles core application functionalities such as internal data structure, object tracking, mapping program and control flow, and

maintaining application states. A key function of this module is to map the question types to an internal data structure, facilitating easy creation and modification of data objects. By default, the application features support for five different types of questions, namely, Mouse Question, Multiple Choice Question, N-Point Question, Keyboard Question and YesNo Question. They will be discussed in more detail in Section 3.6.

More recently, a batch processor module has been integrated into the application. Using this module, the experiment designer can specify parameters that tend to repeat across blocks, trials or questions. The sample test file that is generated through the use of the batch processor can then be opened in the default mode and customized as per the test requirements. This feature is especially useful when the experiment consists of a large number of blocks or trials because it saves the users from having to enter the same information over and over again across different blocks or trials.

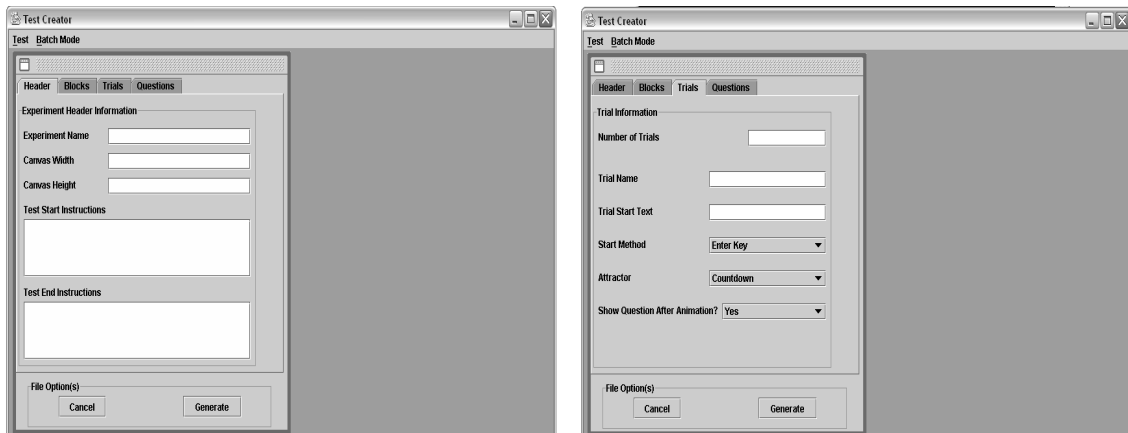


Figure 3.9, Screenshot of the TestCreator module, in batch processing mode.

The file creator module consists of classes associated with generation and processing of files. All experiment files are generated as plain text files. This feature facilitates easy modification of the experiment files without the need for having specialized software. Special care is taken to guarantee interoperability between files

created on different platforms (Unix/Windows). The module is designed to be loosely coupled with other modules, therefore any changes to controller or user interface modules would necessitate little or no changes to the file creator module. Following this approach, the application can be easily extended to work with XML files or Databases.

3.5. FILE FORMAT

Data is transmitted between the different modules through the use of text files. Formatted text file provides the ability to create, view and edit test files without the need for specialized software installation. All files are converted into the standard ASCII Unicode format, to avoid formatting problems between files generated on different platforms. There are four types of files that are used in the application.

- Graphic Files.
- Animation Files
- Log Files.
- Experiment Files.

The following sections will outline the design, layout and format of each of these files.

3.5.1. GRAPHIC AND ANIMATION FILES

Every animation requires at least one object that has to be animated. The graphic file is primarily used by the SKA module to display objects on screen. The file contains specifications for the various objects to be utilized in an animation. In its current toned-down implementation (with fewer features than the full SKA interactive environment), the SKA module is designed to read in graphic files that represent information for a rectangular object.

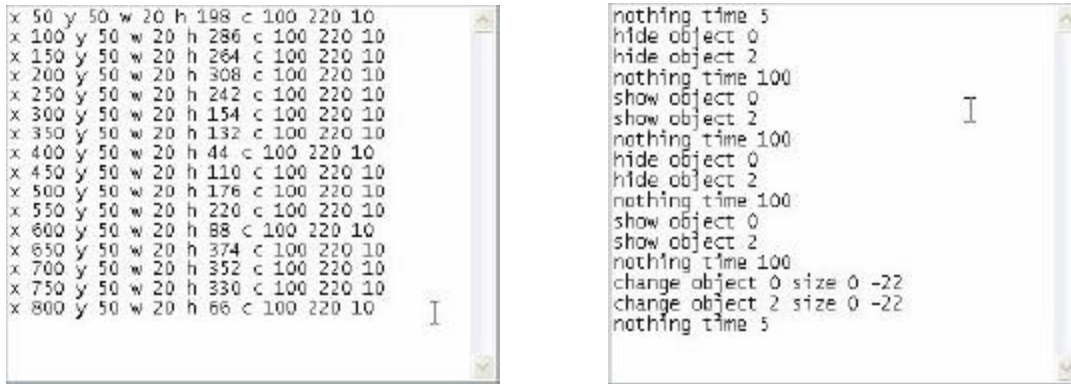


Figure 3.10, Sample graphic and animation files.

Each line in the graphic file represents a single object. The objects are assigned object ID numbers 0 ...n, with the object being represented by the first line of the graphic file assigned the object ID number 0. The format of the graphic file specifies that every line of the graphic file must contain the following attributes and their corresponding values.

- X-Coordinate Value – The value of the X coordinate of the bottom left corner of the object.
- Y-Coordinate Value – The value of the Y coordinate of the bottom left corner of the object.
- Width – The width of the bar (rectangle) in pixels.
- Height – The height of the bar (rectangle) in pixels.
- Color – The RGB values specifying the color of the bar.

The animation file is used by the SKA module to animate the objects specified in the graphic file. The animation file specifies the method and sequence of animation to be performed. Each line of the animation file represents a single command that needs to be executed. Objects are manipulated (animated) by referencing object ID numbers. Hence, every animation command references an object. Some typical animation commands are

delay, show / hide object, change an objects size (length, width), etc. For example, figure 3.10 (right) shows an animation file that makes two objects flash two times.

3.5.2 LOG FILES

Log files are generated by the TestTaker application. The support for automatic logging is an important feature of the TestTaker application. A typical log file contains demographic information of a user, date and time stamp, user responses to questions etc. The log file is designed to be comprehensive, covering all aspects of the experiment. It is intended for use by data analyzers, which are then used to extract data and analyze the results. A typical log file contains the following generic attributes and their corresponding attribute values:

- Date – Represents the date of the experiment.
- Start Time – Represents the time at which the experiment was started.
- End Time – Represents the time at which the experiment concluded.
- Cue Type – Represents the type of cueing employed in the experiment.
- Subject Number – Represents the unique user ID (number) of the test subject.

Apart from the parameters mentioned above, a test file contains other additional attributes and corresponding attribute values pertaining to each trial. This list of attributes is customizable and can be modified depending on experiment needs. They could include, but are not limited to the following attributes:

- Block Number – Represents the block under consideration.
- Trial Number – Represents the trial number under consideration.
- Set Size – The number of bars used in animation (4, 8 or 16).
- Graphic File – The physical location of the graphics file used in the trial.

- Animation File – The physical location of the animation file used in the trial.
- No. Cued – The number of objects which were “cued” in the animation.
- No. Changed – The number of animation which “changed” in the animation.
- Height – The change in height (0 represents no change).
- Actual – Represents whether there was an actual change (true / false).
- Question Number – Represents the question under consideration.
- Correct – Represents of the user response was correct (true / false).

3.5.3 EXPERIMENT FILES

Designing the layout and format of the experiment file was an integral part of the research and a focus of this thesis. The experiment files are generated by the TestCreator Module. This file essentially reflects the overall design of the VizEval application suite.

The file consists of the following parts arranged in a hierarchical order:

- Header – The header supersedes all other information contained within the experiment file. It represents the following attributes associated with an experiment.
 - Experiment name – Represents the name of the experiment.
 - Canvas width - Represents the width (in pixels) of the canvas used for experimentation.
 - Canvas height - Represents the height (in pixels) of the canvas used for experimentation.
 - Number of blocks – Represents the number of blocks within the experiment file.

- Start Instructions – Represents the Instructions to be displayed at the beginning of the experiment.
- End Instructions – Represents the instructions to be displayed at the conclusion of the experiment.

```

TEST_NAME
Vizeval_Experiment_1
GREETING_TEXT
Hello There. Please be relaxed and seated.
ENDING_TEXT
Good Bye and Thank for you time.
NUMBER_OF_BLOCKS
3
CANVAS_WIDTH
800
CANVAS_HEIGHT
400
BLOCK_NUMBER
1
NUMBER_OF_TRIALS
2
BLOCK_START_TEXT
This is the beginning of a block
BLOCK_END_TEXT
This is the end of a block
TRIAL_NUMBER
1
NUMBER_OF_QUESTIONS
5
INSTRUCTION
Answer Question 1-5
ATTRACTOR
Countdown
START_METHOD
spaceBar
GRAPHICS_FILE
16barA.txt
ANIMATION_FILE
4flash1.txt
TRIAL_START_TEXT
Press spaceBar Key
SHOW_QUESTION
true
PRE_ANIM_POST_TIMER_DELAY
4000
QUESTION_TYPE
YesNoQuestion
QUESTION
Did something change?
TIMEOUT_PERIOD

```

Figure 3.11, Section of a sample Experiment test file.

- Block – A block represents a group of trials bound by a common factor. A block represents the following information
 - Block Number – Indicates the relative position of the block within the experiment.

- Number of trials – Represents the number of trials associated with the block.
- Block Start Instructions – Instructions to be displayed at the beginning of the block.
- Block End Instructions – Instructions to be displayed at the end of the block.
- Trial – Trials represent a group of questions that are associated with a particular sequence of animation. A trial has the following parameters associated with it.
 - Trial Name - Represents the name of the trial.
 - Trial Number - The relative position of the trial within its block.
 - Start Text - Instruction to be displayed at the beginning of the trial.
 - Start Method - Represents the manner in which the animations associated with the trial could be started (enter key, space bar, mouse click)
 - Attractor - Represents the attractor that could be used to get the user to focus on screen before the start of the animation, e.g. countdown.
 - Graphic and Animation File - Represents the physical location (on disk) of the graphic and animation files associated with the trial.
 - Post Countdown Pre Animation: This parameter is used to introduce a degree of randomness into the experiment. It represents the delay (in msec) which is introduced after the attractor is displayed but before the animation begins. By default the system selects a value between 150 – 450 mSec for this parameter. However, this can be changed by the user.

- Show Question After Animation: This parameter carries a true/false value. It indicates whether or not the first question must be displayed even before the animation is complete.
- Question – After the completion of animation associated with a trial, a series of questions are displayed to the user. At present, the application supports five different types of questions. Additional question types can be easily integrated into the application suite by extending the base class “Question” to include additional parameters. Each type of question comes with a set of parameters associated with it. The different types of questions and the parameters associated with them are discussed below.
 - Base class “Question” (Keyboard Question) – This type represents those questions that require interaction through the use of a keyboard. The keystrokes following the display of the questions are logged for validation and processing. This class also serves as the base “Question” class from which all other types of questions are extended. The following parameters are associated with this (all) question type(s).
 - Question Text – The text to be displayed.
 - Timeout – The number of seconds for which the question is displayed, after which the next questions is displayed.
 - Mouse Question – This type of question requires interaction through the use of a mouse or a pointing device. It is typically used when the user is required to click on a particular object on screen. In addition to the

parameters extended from the base class (Text and Timeout), this question has the following additional parameters associated with it

- Collect Clicks – If set to true, all mouse clicks on screen following this question are logged (using co-ordinates on screen).
 - Detect Hits – If set to true, the mouse clicks are logged and if any object on screen was clicked upon, the object-Id is also noted in the log file.
- Multiple Choice Question – This type of question has several options one of which is correct. In addition to the parameters extended from the base question class, this question type has the following parameters.
- Number of options – The number of optional choices.
 - Option Values – Values associated with each option.
 - Correct Choice - Represents the correct option.
- NPoint Question – This question type is similar to Multiple Choice Question in the fact that it has several options associated with it. However, unlike multiple choice questions, there is no single correct option. This type of question is used in cases where the answer to a question is based upon an opinion or preference response. e. g. questions involving the use of Likert Scale. In addition to the parameters extended from base class, this question has the following parameters.
- Number of options – The number of optional choices.
 - Option Values – Values associated with each option.

- Yes/No Question – These questions have simple true/false solutions associated with them. This type of question is used when the user response can only be one of two choices true or false. In addition to the parameters extended from the base question class, this question type has the following additional parameters associated with it
 - Correct Choice – Can take values of either true or false.

CHAPTER 4

EXPERIMENT 1: A PILOT STUDY

4.1. AIM

The aim of this experiment is to test the functionality of the VizEval software suite and to gather some preliminary data for the design of forthcoming experiments. A principal goal of this experiment will be to study the effects of cueing on human perception of changes in an animation. We begin with the hypothesis that cueing has an effect on the ability to detect “just noticeable” changes in the height of graphical objects.

4.2. EXPERIMENTAL SETUP.

The pilot study involved five graduate students. The experiment was conducted on Dell Dimension 8300 PC's with 17 inch flat screen LCD monitors. The participants were placed 50 cm away from the center of the screen while performing the experiment so that the display would be within the useful field of view.

The graphical objects used in the animations were rectangular in shape with 20 pixel widths and varying heights. The cue type employed was “flash”. The height of the rectangle range from 44 pixels to 374 pixels increasing or decreasing at 22 pixel intervals, the reason being that 22 pixels has been found to be the “just noticeable difference” in height change (JND: the minimum value for a change in height that can be recognized when a focal point in being attended to). Pilot studies indicated that the optimal value for the JND is 22 pixels. The experiment included animations that contain four, eight or sixteen bars. Pilot studies conducted at the Dr. Davis labs, Georgia Institute

of Technology, indicate that green color has the most impact on perception, hence the color of the bars used were green, with RGB values of 100, 200 and 10 respectively.

The animation files were designed to test the effect of cueing on the detection of “just noticeable” changes in the height of the bars. Given any trial, one or two bars were cued and zero, one or two bars were changed. The figure below represents the possible combinations of cues and changes that were employed.

		Change		
		0	1	2
C u e	1			
	2			

Figure 4.1, Pictorial representation of Cue-Change combinations

Pilot studies also indicated that most humans can recognize simultaneous cues and changes in two objects. Increasing the number of objects changed or cued beyond two could be over burdening for some test subjects, challenging the quality of feedback collected. Thus, it was decided that at the most two objects will be cued or changed in any given animation.

At the beginning of the experiment, participants are instructed to use a countdown timer to act as a focal point throughout the experiment. Before any animation can begin, the timer counts down to zero. The countdown timer, set at value zero, remains visible throughout the sequence of animation and disappears once the animation is complete. The bars (objects) are placed symmetrically from the edge of the canvas across the useful field of view. In an experiment involving four bars, two bars are placed in the leftmost

position and two other bars are placed at the rightmost position. The same concept is applied to eight and sixteen bars, respectively.

The experiment has a total of 288 trials spread across ten blocks with five questions per trial, testing all combinations of graphics and animation files. Each trial contained five questions about the animation, they are:

1. Type: YesNO, Did any of the bars change height?
2. Type: Mouse, Please click on the bar most likely to have changed height.
3. Type: NPoint, How confident are you that this bar actually changed height?
4. Type: Mouse, Please click on the bar second most likely to have changed height.
5. Type: NPoint, How confident are you that this bar actually changed height?

4.3 DISCUSSION

The data collected is being used to evaluate factors such as the effect of cueing on helping users notice a change, and the effect of location of a change within the field of view to the user's ability to detect a change. Also, the data collected from this experiment will be combined with results of future studies to compare the effects of various cueing strategies. In the study, participants were asked to complete 72 trials, a subset of 288 trials.

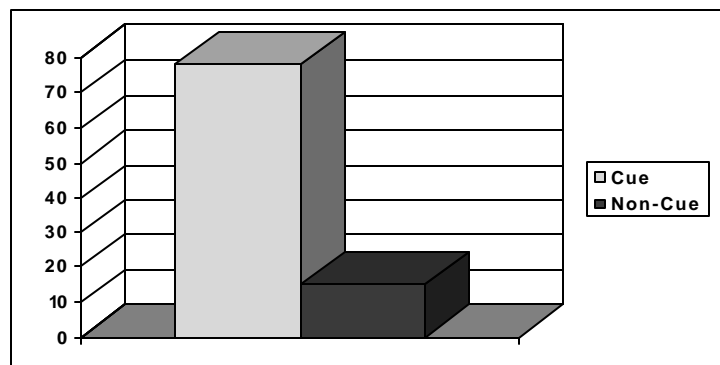


Figure 4.2, Graphical representation of results.

The results suggest that participants correctly responded to 78% of the bars that changed height when they were cued as compared to only 15% that changed height when the bars were not cued. The data shows that a majority of the participants responded more accurately to the bars that were cued prior to changing their heights. The results also suggest that the participants generally did not recognize changes in bars that were not cued. This preliminary analysis is based on the study involving a small group of participants. A larger study involving more than one hundred undergraduate students is being conducted at the Georgia Institute of Technology. The results from that study will permit an assessment of the statistical significance of this result.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Through the pilot study, it is safe to assume that the VizEval suite permits researchers to conduct experiments on the effectiveness of program visualization. For example, the results of the pilot study suggest that cueing does have an effect on human perception of noticing whether or not a bar changed in its height. The bars that were cued were more likely to be detected than those that were not cued. Thus, the initial hypothesis that cueing has an effect on human perception of detecting changes to an object is true.

Specific to the TestCreator, the pilot study was designed and the test files were generated in under eighty eight minutes. To code a similar test file by hand would have taken considerably longer time. Furthermore the chances of a syntactic error would have been very high, rendering the generated file useless. TestCreator could be effectively used to generate fast and error-free test files specific to the VizEval experimental platform. Furthermore, the functionalities incorporated into the batch-mode of the TestCreator would typically reduce the time taken to generate a similar file by half.

5.2 FUTURE WORK

“People only see what they are prepared to see”

-Ralph Waldo Emerson, Journals, 1863

During the past twenty years a large volume of research has been directed at the use of computers and technology in classrooms. Many animation tools have been created

to facilitate teaching of algorithms in classrooms [26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. However, as discussed in this thesis, a large majority of the research has insufficient attention to the perceptual factors. As a result, studies have produced mixed results. Through VizEval we seek to identify and evaluate the factors that contribute to the effectiveness of program visualizations.

In many ways VizEval has addressed the need for a comprehensive testing platform for conducting experiments in perceptual psychology and ultimately, algorithm animations. Yet, a large volume of work remains to be done to improvise the tool.

Specific to this thesis, the TestTaker application must be improvised to address the following factors

- At present, there is no mechanism to view an animation within the TestCreator. This poses a challenge while designing experiments because the experiment architects have to use the SKA animation package to view the animations and subsequently formulate the questions pertaining to the animation.
- With the inclusion of the batch processing module into the TestCreator, designing large and complex experiments have become much easier. However in its present implementation, the batch processor supports only a small set of parameters which tend to repeat across many blocks of trials. This feature needs to be extended to include multiple sets of parameters which could be assigned to trials in a drag and drop fashion.
- Integrating the FileCreator application into the TestCreator application would facilitate a solution to create and modify graphic and animation files on the fly.

The above mentioned list of possible changes to the application is by no means comprehensive. During the course of designing the application, a large number of parameters have been identified and support for the same has been included into the application. It is quite possible that experiments in the future would include attributes not discovered so far, necessitating further modifications.

BIBLIOGRAPHY

- [1] <http://webster.cs.uga.edu/~eileen/VizEval/>
- [2] Bartram, L. Enhancing Information Visualization with Motion. Unpublished Ph.D. thesis, School of Computing Science, Simon Fraser University, Canada, 2001. <http://fas.sfu.ca/pub/cs/theses/2001/LynBartramPhD.pdf>
- [3] Bartram, L., Ware, C., and Calvert, T., Moving Icons: Detection and Distraction. In proceeding of Interact 2001, Tokyo, Japan.
- [4] Z. W. Pylyshyn and C.R. Sears. Multiple Object Tracking and Antinational Processing. Canadian journal of Experimental Psychology, 2000, 54(1), 1-14.
- [5] Byrne, M., Catrambone, R., Stasko, J., Do Algorithms Aid Learning? Technical Report GIT-GVU-96-18, August 1996
- [6] Lawrence, A., Badre, A., Stasko, J., Emperically Evaluating the Use of Animations to Teach Algorithms. Technical Report GIT-GVU-94-07.
- [7] Kehoe, C., Stasko, J., Taylor, A., Rethinking the evaluation of algorithm animations as learning aids: an observational study. International Journal of Human-Computer Studies (2001) 54, 265-284.
- [8] Faraday, P., Sutcliffe, A., An Empirical study of Attending and Comprehending Multimedia Presentations. CHI 97.
- [9] Ross, R. A Testing Environment for the Evaluation of Program Visualization. Unpublished Thesis, Department of Computer Science, The University of Georgia, USA, 2004.
- [10] Hansen, S., Schrimsher, D., Hegarty, M., Narayanan, H., Emperical Studies of Animation-embedded Hypermedia Algorithm Visualizations. Technical Report CSE98-06, Auburn University.
- [11] Healey, C., Amant, R., Elhaddad, M., ViA: A perceptual Visualization Assistant, Dpeartment of Computer Science, University of North Carolina.
- [12] Taylor, A., Kraemer, E., SKA: Supporting Algorithm and Data Structure Discussion. Computer Science Department, University of Georgia.

- [13] Taylor, A., Kraemer, E., Tudoreanu, M., Why Johnny Wont Visualize. University of Georgia, Department of Computer Science & Washington University, St. Louis, MO, Department of Computer Science.
- [14] McCormick, B.H. et. al. (ed), Visualization in Scientific Computing, Computer Graphics21(6), November 1987.
- [15] Denning, P., "Computing, Applications, and Computational Science", Communications of the ACM, 34(10), p. 129, October, 1991.
- [16] Owen, S., "Visualization Education in the U.S.A", Journal of Computers and Education, Vol. 8, pp. 339-345, 1993.
- [17] Denning, P.J, A debate on Teaching Computer Science. Communications of the ACM, c32 n.12, pp. 1397-1414, December 1989.
- [18] Stasko, J., Badre, A., Lewis, C., Do algorithm animations assist learning? An empirical study and analysis. In proceedings of the ACM INTERCHI' 93 Conference on Human Factors in Computing Systems. Understanding Programming, pages 61 – 66, 1993.
- [19] B. Price (1990) A framework for the automatic animation of concurrent programs. Unpublished M.S. thesis, Department of Computer Science, University of Toronto.
- [20] M. E. Crosby& J. Stelovsky (1995) From multimedia instruction to multimedia evaluation. Journal of Educational Multimedia and Hypermedia 4, 147 - 162.
- [21] J. S. Gurka (1996) Pedagogic Aspects of Algorithm Animation. Unpublished Ph.D. dissertation, Computer Science, University of Colorado.
- [22] C. Kann, R.W. Lindeman,&R.Heller (1997) Integrating algorithm animation into a learning environment. Computers&Education 28, 223 -228.
- [23] P.Mulholland (1998) A principled approach to the evaluation of SV: a case study in Prolog. The MIT Press, Cambridge, MA, pp. 439- 452.
- [24] C. D. Hundhausen, & S. A. Douglas (2000)Using visualizations to learn algorithms: should students construct their own, or view an expert's? In: Proceedings 2000 IEEE International Symposium on Visual Languages. IEEE Computer Society Press, Los Alamitos, pp. 21-28.
- [25] D. J. Jarc, M. B. Feldman & R. S. Heller (2000) Assessing the benefits of interactive prediction using web-based algorithm animation courseware. In: Proceedings SIGCSE 2000. ACM Press, New York, pp. 377-381.

- [26] Stasko, POLKA - Technical Report, <ftp://ftp.cc.gatech.edu/pub/gvu/tech-reports/92-10.ps.Z>
- [27] Stasko, SAMBA Animation Designers Package, From the official SAMBA website at the SV website: <http://www.ads.tuwien.ac.at/teaching/ss03/186084/sambadoc.pdf>
- [28] Taylor, A, Kraemer, E. SKA: supporting algorithm and data structure discussion, ACM Press, 33rd SIGCSE (2002), p58 – 62
- [29] Robling, G., Friedleben, B. Approaches for Generating Animations for Lectures, Department of Computer Science, University of Seigen, Germany.
- [30] Ari Korhonen, Lauri Malmi, Panu Silvasti, Ville Karavirta, Jan Lönnberg, Jussi Nikander, Kimmo Stålnacke, and Petri Tenhunen: Matrix - A Framework for Interactive Software Visualization.
- [31] W. Pierson and S. H. Rodger, Web-based Animation of Data Structures Using JAWAA, Twenty-ninth SIGCSE Technical Symposium on Computer Science Education, p. 267-271, 1998.
- [32] Algorithm Animation at Compaq Research Center, SRC. <http://research.compaq.com/SRC/zeus/home.html>
- [33] Brown. M, An introduction to Zeus: audio visualization of some elementary sequential and parallel sorting algorithms. SIGCHI Conference, ACM Press. 1992.
- [34] Stasko, John T., "Animating Algorithms with XTANGO", SIGACT News, Vol. 23, No. 2, Spring 1992, pp. 67-71.
- [35] Marc H. Brown and Marc A. Najork. Collaborative Active Textbooks. Journal of Visual Languages and Computing, 8(4):453-486, August 1997.
- [36] <http://www.pstnet.com/products/e-prime/>
- [37] Narayanan. H, Hansen, S. On the role of Animated Analogies in Algorithm Visualizations. Fourth International Conference of the Learning Sciences (pp. 205 – 211).
- [38] LogoMedia – DiGiano, 1992 at the University of Toronto. Associates program events with non-speech audio. The sounds are generated using a MIDI synthesizer and are when a specific event occurs.
- [39] Cox, Kenneth and Gruia-Catalin Roman (1993). A Taxonomy of Program Visualization Systems IEEE Computer.
- [40] Kraemer, Eileen and John T. Stasko (1993). The Visualization of Parallel Systems: An Overview. Journal of Parallel and Distributed Computing 18, 105 – 117.

[41] Myers, B. A. (1990). Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages and Computing* 1(1): 97-123.

[42] NSF Award # 0308063. Program Visualization: Using Perceptual and Cognitive Concepts to Quantify Quality, Support Instruction and Improve Interactions. June 15th 2003 – May 31, 2005. Principal Investigator: Eileen Kraemer.
