

A TESTING ENVIRONMENT FOR THE EVALUATION OF PROGRAM VISUALIZATION QUALITY

by

MATTHEW ROSS

(Under the direction of Eileen Kraemer)

ABSTRACT

Program visualizations have the potential to convey information about the behavior of the programs they depict. However, program visualizations in use vary widely in their ability to convey the desired information. In this work we conduct a study of the “quality” of the visualizations, the ability of those visualizations to present information in a way that is both effective and efficient in creating understanding. In order to measure program visualization quality, many traits of the visualization need to be considered: size, shape, color, and location of graphical elements, cueing and motion styles, etc. A testing environment called TestTaker has been developed that allows the experimenter to display visualizations to the user and receive feedback. The feedback is then used to evaluate the quality of the program visualizations based on the metrics specified. These “quality” metrics will be used to create a standard way of measuring the quality of program visualizations.

INDEX WORDS: Program Visualizations, Metrics, Quality

A TESTING ENVIRONMENT FOR THE EVALUATION OF PROGRAM VISUALIZATION
QUALITY

by

MATTHEW ROSS

B.S., Saint Peter's College, 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

Matthew Ross

All Rights Reserved

A TESTING ENVIRONMENT FOR THE EVALUATION OF PROGRAM VISUALIZATION
QUALITY

by

MATTHEW ROSS

Major Professor: Eileen Kraemer

Committee: Dan Everett
Maria Hybinette

Electronic Version Approved:
Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2004

TABLE OF CONTENTS

LIST OF FIGURES	vi
CHAPTER	
1 INTRODUCTION	1
1.1 VISUALIZATION UNDERSTANDING	2
1.2 PURPOSE	4
2 BACKGROUND AND RELATED WORK	6
2.1 PERCEPTUAL PSYCHOLOGY	
2.1.1 THE PTUAL MODEL	6
2.1.2 SELECTIVE ATTENTION	8
2.1.3 DIVIDED ATTENTION	9
2.2 EVALUATION OF VISUAL DISPLAYS	10
2.2.1 SEARS AND PYLYSHYN	10
2.2.2 POLKA AND SAMBA	11
2.2.3 E-PRIME	11
3 VIZEVAL SUITE	13
3.1 SUPPORT KIT FOR ANIMATION (SKA)	14
3.2 FILECREATOR	14
3.3 TESTCREATOR	15
3.4 TESTTAKER	17

4	EXPERIMENT	22
4.1	INTRODUCTION.....	22
4.2	SETUP	22
4.3	DISCUSSION	25
5	CONCLUSION AND FUTURE WORK	26
5.1	CONCLUSION	26
5.2	FUTURE WORK	26
	BIBLIOGRAPHY	28

LIST OF FIGURES

3.1 VizEval Suite Architecture	13
3.2 Sample Graphics File	15
3.3 Sample Animation File	15
3.4 Sample Test File	16
3.5 <i>TestTaker</i> Introductory Screen	18
3.6 <i>TestTaker</i> Instruction Screens	19
3.7 <i>TestTaker</i> Main Interface	20
4.1 Cue-Change comparision	23
4.2 Sample 4 Bar Graphics File Placement	24
4.3 Table of Results for Experiment 1	

CHAPTER 1

INTRODUCTION

A substantial body of research exists in the use of visualizations to convey information about programs. The desire to visualize comes from a strong intuitive belief that visualization is valuable for communicating the state and behavior of programs, not only in academic settings, but whenever people want to convey information about the behavior of dynamic systems. In particular, in the area of algorithms and data structures, traditionally regarded as a core area of computer science that students find difficult [6], the advent of algorithm animation created great expectations in the CS teaching community, as animation was seen as a better way to portray the dynamic workings of algorithms than was the use of purely static media.

However, a disconnect exists between the belief in how useful visualizations are and the extent to which those visualizations are actually employed. One factor in the underutilization of program visualization may be doubts about how beneficial the visualizations can be for instructional use. Although some studies have shown benefits for the use of algorithm animation for instructional purposes, the overall results have not been convincing on either side, and researchers have not yet fully explained why the results of many studies have been inconclusive.

Beyond this concern about the ultimate benefit, a number of other factors may contribute to the failure of visualization to meet the expectations of the CS teaching community and to be widely employed. Factors that may inhibit utilization of program visualization include questions about the benefits of visualization to understanding, and the effort required for visualization

creation and visualization navigation. While all three are important factors that contribute to the usefulness of program visualizations, this thesis focuses only visualization understanding.

1.1 VISUALIZATION UNDERSTANDING

One problem that limits the usefulness of program visualization is that viewers may have difficulty in understanding the message that the designer of the visualization is trying to convey. A number of studies of algorithm animations have been done in the past decade, with mixed results [1, 4]. A largely unacknowledged complicating factor in these studies is the "goodness" of the visualizations themselves: Are people able to perceive what the visualization designers intended? Are users able to comprehend the message that the designer was trying to convey? If not, what is the reason? Viewers may perceive objects on the screen and map these individual objects and their properties to program entities and their properties but fail to discern the patterns in the relationships among elements, perhaps because the number of elements is too great, or the relationships too complex, or because of apparent conflicts or contradictions in the various mappings that have been applied.

Researchers studying the benefits of program visualization have not previously considered perceptual factors in their studies. A study, entitled *Program Visualization: Using Perceptual and Cognitive Concepts to Quantify Quality, Support Instruction and Improve Interactions*, funded by the National Science Foundation and conducted in part at the Kraemer Lab at the University of Georgia [19], seeks to answer many important questions asked about the processes of perceptual tracking, attentional processing, and conceptual mapping when viewers watch an animation:

1. Are viewers able to clearly perceive objects on the screen (distinguish individual objects, determine color, size, pattern of motion, determine number of objects, relationships between moving objects)?

2. Are viewers able to map objects on the screen and their properties onto the program entities and their properties?

3. Are viewers able to correlate events in the animation with events in the algorithm?
What is the effect of speed/pacing?

4. Is prior or concurrent guidance (in the form of audio explanations, text, visual cues) helpful in interpreting visual phrases?

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

At present PV systems have largely ignored the issue of appropriate perceptual properties for effective viewing. Effective PV systems must support perceptually appropriate animation, graphical design and layout, as well as good pedagogical design. In order to address this problem, we seek to evaluate perceptual and attentional aspects of program visualization. Goals of the overall project will be to:

1. Investigate characteristics of perceptual tracking, attentional processing, and conceptual mapping relevant to program visualization.

2. Gain insight into the nature and effects of perceptual limitations relevant to visualization of processes, with particular emphasis on program visualization.

3. Gain insight into the effects of perceptual limitations in the face of varying levels or types of cognitive load.

4. Evaluate the efficacy of existing techniques for use in visualizations.

5. Define quality metrics for PV.
6. Develop design guidelines that apply specifically to program visualization.
7. Develop a 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.
8. Produce an evaluation matrix of the intersection of information type with visualization technique.
9. Propose how these guidelines might be applied to visualization in general.

1.2 PURPOSE

The focus of this thesis is to create an environment in which such perceptual, attentional, and conceptual studies can be conducted. All of the previous experiments and projects used program visualizations that were created based on what the experimenters thought were good parameters. In actuality, the design may have been flawed by use of a program visualization that did not convey the best meaning or provide the best criteria for evaluation. Our goal is to go back to the basics and evaluate visualizations, from the most simple to the most complex, based on various attributes, such as size, shape, color, orientation, position of graphical objects, and their motion. We will attempt to collect empirical data that will eventually define a system of metrics that quantify the quality of a visualization and its ability to convey information. By developing a standard system of metrics, visualizations can be more uniformly evaluated, design improved, and program visualization accepted by a larger population.

A testing environment has been developed that will begin to address these issues. The environment that has been developed is called the **VizEval Suite**. It is designed for the creation and evaluation of program visualizations. As part of this thesis, a single part of the VizEval Suite was developed, known as *TestTaker*. *TestTaker* was designed to be the testing

environment for the *VizEval* suite. Its purpose is to provide an environment in which the study can be automated. It performs this task by executing a test file that is created by another application in the VizEval Suite, the *TestCreator*. While running the test, experimental data is collected automatically as the participants navigate through the testing environment. The benefits of automated collection allow for an increase in the number of participants being tested, with multiple tests being implemented simultaneously on separate workstations. It also allows a greater ease in collecting and analyzing data. The entire VizEval Suite is written in Java, giving it platform independence.

CHAPTER 2

BACKGROUND AND RELATED WORK

Our study of program visualization effectiveness relies on prior work in perceptual psychology, empirical studies of program visualization and the development of experimental test systems and suites.

2.1 PERCEPTUAL PSYCHOLOGY

In order to fully understand how we interpret program visualizations and what characteristics contribute to the “quality” of a visualization we look to the field of perceptual psychology. Looking at studies from this area provides insight and information about how people perceive visualizations and how they process the information that they perceive. What is important from these studies are the conclusions that are drawn from the data collected through the TestTaker application. We focus on the areas of the perceptual model and attention, specifically selective attention and divided attention. These areas provide important background information that will help foster an understanding of the architecture and design of the TestTaker application.

2.1.1 THE PERCEPTUAL MODEL

In the perceptual model, processing is divided into two stages. Stage one is parallel processing of the entire visual scene. In other words, the entire image that is captured with our eyes is processed and what are considered low-level properties are extracted in parallel. Stage two deals with higher-end processing that allows us to recognize what we see. This is where object recognition and memory recall occur [14].

Stage1: Extraction of low-level properties in parallel

When a visual scene is first processed by the eye, various information is collected in parallel. The eye is tuned to pick up assorted differences in the characteristics of the scene. These are low-level properties that can be discerned without knowing exactly what is contained in the scene. For example, when looking at a room where a board meeting takes place, one would first notice that there are various edges and corners in the room, as well as colors and textures. This information is what is first picked up by the eye and transferred to the brain. These low-level properties do not give us a great deal of information about the image or object that we see. What it does give us is a sense of structure about the scene. We do not make the decision that it is a room filled with chairs and a table until Stage 2.

Stage 2: Object recognition and memory recall in sequence

Now that we have collected the underlying basic characteristics of the scene, it is time for our brain to recognize what we see. This process is done in sequence, as we can only recognize one object at a time. It would be impossible to recognize the table and the chairs at the same time because one must be able to make the distinction between the table and the chairs, because they share certain edges. Either the edges form a table first, after which you can recognize the fact that there are chairs at the table, or vice versa. Either way, you have to recognize an object and its boundaries before recognizing more objects. One theory about how we recognize objects is the Geon Theory, which states that the visual system perceives 3D objects as being made up of separate 3D component objects called geons [14].

Object recognition relies on recall from memory as to what objects have been previously encountered and classified by the viewer. This involves retrieving objects in memory and comparing them to the object that is being viewed. Both short-term and long-term memory are

utilized depending on the nature of the task. Short-term memory is where something is placed before it is stored in long-term memory. However, short-term memory only allows the storage of 4-7 items before it is considered full [12]. Any more than this will replace an object in short-term memory with a new object. After a few seconds the objects in memory are moved into long-term storage. However, retrieval from long-term memory is only as good as the recall reference that was used to store it. If the references to the object are strong, then it will be no problem to recall. Things such as repetitive use or common interaction are stronger references than one time occurrences.

We now turn to a different aspect of perceptual psychology, which focuses on attention. This is still a heavily debated topic in the field of psychology. However, it has important applications when dealing with program visualizations, in particular, those visualizations that are used for instruction. In the course of discussing the relevancy of studying attention from a perceptual psychology perspective, we will focus on attention from two different angles: selective attention and divided attention.

2.1.2 SELECTIVE ATTENTION

Selective attention is the process of focusing on a single input stimulus, whether it be aural, visual, or sensory (touch) [12]. A common real-world example of selective attention would be holding a conversation with a person in a room full of people talking at the same time. You filter out the other voices and selectively attend to the voice of the person with whom you are talking. While our focus is not on selective hearing, it provides a basic example of what selective attention is about. Selective visual attention is slightly more complicated, because where our eyes are looking towards may not be where our attention is focused [10]. In general, however, where we look is what we attend to. For example, when reading a book, or say this

paper, you are selectively reading the words on the page. However, at any given time, motion in the periphery, perhaps a person's movement, for instance, may draw your attention away from the book without you lifting your eyes and looking at the source of the motion. When your attention is once again focused on the book you find that you have no idea where you last left off reading, even though you were still looking at the page.

Another interesting aspect of selective visual attention is the fact that it is impossible to look in any direction that you have not already given attention to [12]. You must first attend to a direction and then you can look in the direction you are attending. For example, you are unable to look up without first focusing your attention up. This attribute of selective attention is a part of what TestTaker is designed to evaluate, which will be discussed later in the paper.

2.1.3 DIVIDED ATTENTION

Divided Attention is the process of focusing on multiple input stimuli and the ability of the perceptual system to handle those inputs [12]. This is a more realistic depiction of everyday practice by people. For example, driving a car is an example of processing multiple visual inputs. While driving you must keep your eyes on the road, yet maintain inputs from your mirrors, traffic signals, other vehicles, the instrument panel, etc. The question to answer then is: What are the thresholds of the visual perception system in terms of multiple inputs? To put it another way, are there limitations to the visual perception system, and if so, what are they?

When dealing with the experiments to test the visual system, how are people to know what they should attend to? This can be answered in different ways. One way is to predefine the locations that should be attended to. This has had serious drawbacks in that when told where to attend, the subjects did not attend anywhere else and thus missed other stimuli. A more effective way that provides better results is to use cues. A cue is a form of stimulus that is used to signal

that an event is going to take place. The form of the cue can vary as well, with the most common ones being aural or visual. In the case of visual cues, there are many different ways to signal the user to attend to a change, such as change in color, flashing, pulsating, wiggling, motion, etc [2]. Some of these cues have been implemented in TestTaker and are being used in developing an evaluation of effectiveness of cueing strategies as an element of the metrics we are developing.

2.2 EVALUATION OF VISUAL DISPLAYS

Some researchers are trying to evaluate stimulus and response in order to gain some measurement of the visual systems capacity and limitations [3]. Empirical studies have examined this question [15], and various applications have been developed to explore this in an automated manner [13]. What follows is a brief discussion on a few of these studies and projects, their similarities and differences, and how they are related to the VizEval Suite.

2.2.1 SEARS AND PYLYSHYN

Sears and Pylyshyn conducted a study designed to explore the limitations of the visual attention system in tracking multiple objects in motion [15]. Previous experiments revealed that the visual system can index and track about 4-5 objects in motion at any given time. However, Sears and Pylyshyn focused specifically on the relationship between visual indexing and attentional processing. In the experiment, there were target objects that were cued with a flash and distractor objects that were not cued. When the trial began, all the objects began to move, but the participants were supposed to keep track of the targets that were cued, while maintaining a fixation on a specific spot on the screen, noted by a cross. During the trial, one of the targets or the distractors changed form, and that was their cue to press a button to stop the trial. The results showed that participants responded faster to target changes than to distractor changes. The

results from this experiment were used as an aid in developing Experiment 1 for the VizEval Suite application.

2.2.2 POLKA AND SAMBA

Polka is a visualization system developed at Georgia Tech that allows the user to view animations of algorithms and programs [16]. Its chief focus is providing the ability to view animations of algorithms and programs that run in parallel. Also included with Polka is Samba, which is a front-end animation interpreter. Samba was designed to execute regular ASCII commands [17]. There is one command per line, with the animation being performed after each line is read in. The development of Samba was to allow the animation files to be readable by any program, allowing for ease of transition for the application into multiple programming language platforms. This idea is similar to what is trying to be accomplished by the VizEval Suite; however, our focus is not on the algorithms themselves, but rather what is being used to visually depict the behavior of the algorithms and whether or not it is the best method to do so.

2.2.3 E-PRIME

E-PRIME is a suite of applications that is comparable to VizEval Suite. E-PRIME allows the experimenter to develop an experiment, implement it, collect the data, and then analyze it [18]. However, the major difference between E-PRIME and VizEval Suite is what the tests are designed to do. VizEval Suite is designed to evaluate the attributes of program visualizations in order to develop a system of metrics for “quality of visualizations”, which could have potential influence over which visualizations are used in future psychological experiments. E-PRIME is developed to conduct psychological experiments in an automated environment. The core system of E-PRIME includes: E-Studio which is used to create the experiments, E-Basic which is the underlying scripting language, E-Run which is used to run the experiment, E-Merge which

allows combining of single session experiments, E-DataAid which gives security features and the ability to edit and export data, and E-Recovery, which allows the recovery of data from an unexpected termination or a corrupted file. VizEval Suite and E-PRIME are similar in terms of underlying design, in that both are broken down into several components that allow for ease of use. E-PRIME, however, implements its run-time environment in E-Basic, a language that is specific to E-PRIME. This will limit the usability in terms of platform independence.

CHAPTER 3

VIZEVAL SUITE

VizEval Suite is an application that has been developed for the purpose of empirically evaluating the effects of properties of program visualizations on viewer comprehension. It allows the experimenter to develop an experiment from scratch, including the specification of all of the graphics, animations, and questions that will be utilized. VizEval suite can be broken down into several components: the *Support Kit for Animation(SKA)*, *FileCreator*, *TestCreator*, and *TestTaker*. Figure 3.1 represents the system architecture of VizEval Suite. *FileCreator* creates the graphics and animation files. *TestCreator* uses the graphic and animation files to create the test files. *TestTaker* uses the test files to run an experiment and *SKA* to display the graphics files and execute the animation files.

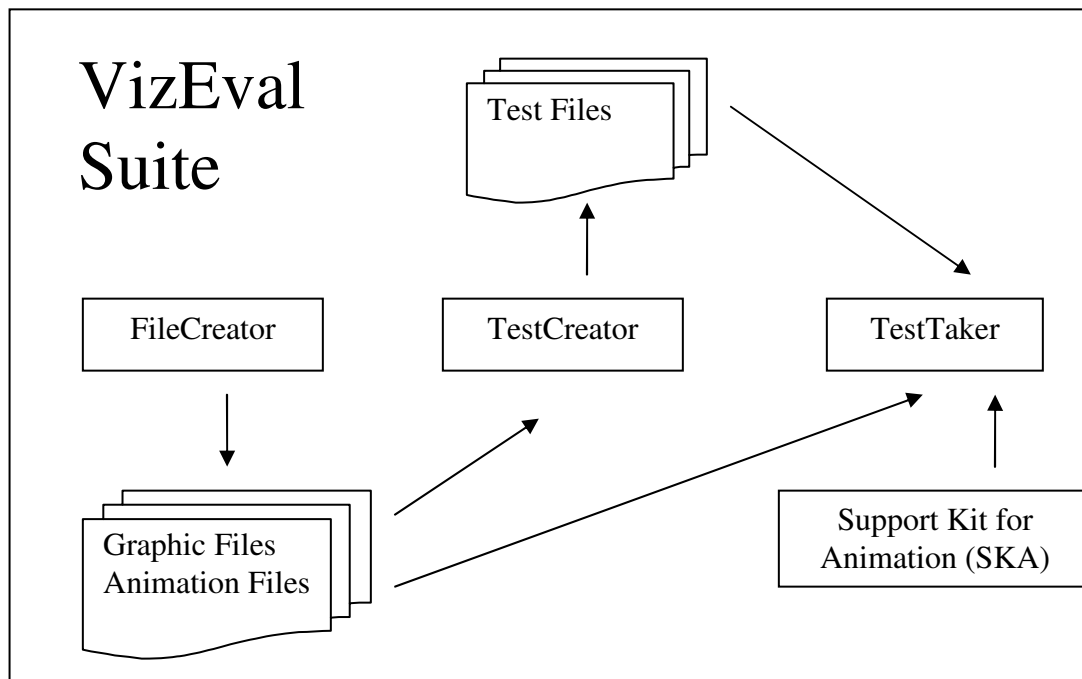


Figure 3.1: VizEval Suite Architecture

3.1 SUPPORT KIT FOR ANIMATION (SKA)

The *Support Kit for Animation* is the background application that controls both the graphics that are displayed and the animations that occur. The graphics are displayed by using a canvas onto which objects are painted. The *SKA* tool allows for the creation of different basic visualizations such as lines, arcs, circles, rectangles and squares. More complex visualizations can be constructed from these. The objects are specified in the graphics file, which gives a set of x and y coordinates, measurements for width and height, and three values for color using the RGB color scale with values ranging from 0-256. Objects may be placed anywhere on the canvas, arbitrary shapes can be formed, and any color that can be achieved based on the 256(0-255) value RGB color scale may be specified.

Objects are animated by means of modifying the x and y coordinates of the object by a certain interval and repainting the canvas so that the object appears in the newly specified coordinates. This gives *SKA* the ability to have smooth movement if desired or short choppy motions depending on the needs of the experimenter. The animation sequence is written to an animation file that specifies an action, the object that the action is being acted upon, and the new parameters for the specified action. The types of action can be movement, change in color or height, or having the object disappear or reappear. The animation is performed by an animation engine that interprets animation files written in the specified format. The engine stores the sequence of operations from the file in an array and performs them sequentially. This allows the animations to be done in real-time.

3.2 FILECREATOR

The purpose of *FileCreator* is to create the graphics and animation files that are used in the experiment. Currently, *FileCreator* is configured to create all of the necessary graphic and

animation files for Experiment 1, which will be discussed in greater detail in Chapter 4. For experiment 1 we selected bars (rectangles) as the graphical object. The interface for *FileCreator* allows the experimenter to select bars according to a pre-specified set of 4, 8, or 16. The experimenter can set the size of each bar in the groups of 4 or 8, but the group of 16 has all bar heights that are used. After creating the bars and specifying the bar heights, the graphics files are created. After that, the experimenter has the option to create animation files based on the newly created graphics files. The experimenter also has the option to manually select which bars to animate and which bars to cue. Once finished, the animation file is created.

```
x 50 y 50 w 20 h 198 c 100 220 10
x 100 y 50 w 20 h 286 c 100 220 10
x 150 y 50 w 20 h 264 c 100 220 10
x 200 y 50 w 20 h 308 c 100 220 10
x 250 y 50 w 20 h 242 c 100 220 10
x 300 y 50 w 20 h 154 c 100 220 10
x 350 y 50 w 20 h 132 c 100 220 10
x 400 y 50 w 20 h 44 c 100 220 10
x 450 y 50 w 20 h 110 c 100 220 10
x 500 y 50 w 20 h 176 c 100 220 10
x 550 y 50 w 20 h 220 c 100 220 10
x 600 y 50 w 20 h 88 c 100 220 10
x 650 y 50 w 20 h 374 c 100 220 10
x 700 y 50 w 20 h 352 c 100 220 10
x 750 y 50 w 20 h 330 c 100 220 10
x 800 y 50 w 20 h 66 c 100 220 10
```

Figure 3.2: Sample Graphics File

```
nothing time 5
hide object 0
hide object 2
nothing time 100
show object 0
show object 2
nothing time 100
hide object 0
hide object 2
nothing time 100
show object 0
show object 2
nothing time 100
change object 0 size 0 -22
change object 2 size 0 -22
nothing time 5
```

Figure 3.3: Sample Animation File

3.3 TESTCREATOR

TestCreator is used to specify the series of trials that will be used in the experiment. It allows the experimenter to control various aspects of the tests such as time delays, animations, graphics, questions, answers, question types, etc. The experimenter builds the test based on the specific attribute of the visualization being evaluated. In some cases, the actual visualization itself is being evaluated, and in other cases, the various cues and timing delays are being evaluated. An experiment consists of sections of blocks, trials, and questions. Each block contains a set of trials and each trial contains a set of questions. Also associated with each trial is

a graphic file and an animation file, which are obtained from the set of graphics and animation files that were created using *FileCreator*. The experimenter enters the questions that are associated with each animation and the appropriate responses for each of the questions based on the experimenters criteria.

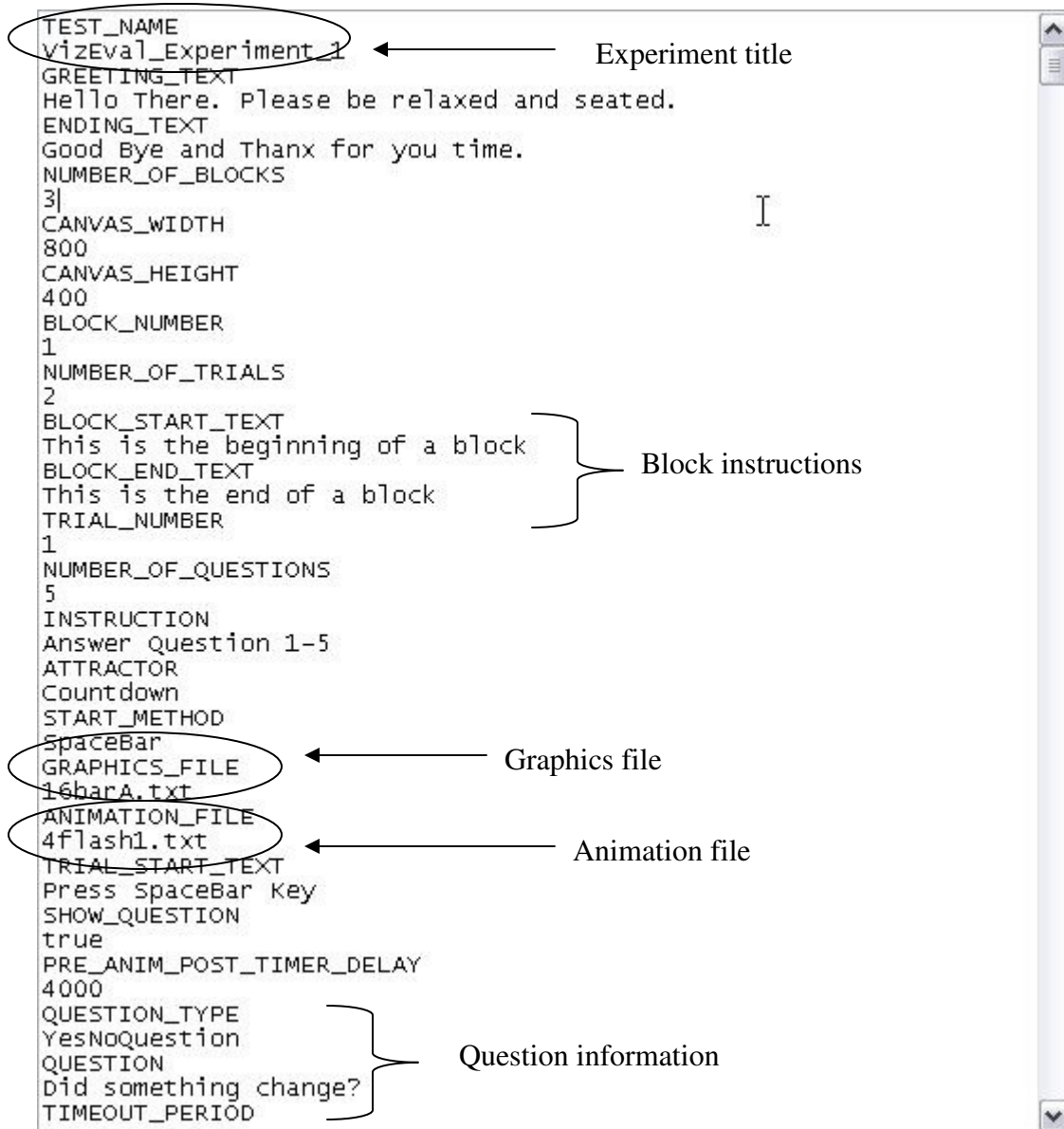


Figure 3.4: Sample Test File

There are many types of questions that can be used. A key board question means that the input response must be entered from the keyboard. A multiple choice question means that the

response must be made from a set of pre-determined choices which are specified by the experimenter. A click question means that the input response must be from the mouse click and is usually linked to clicking an object on the canvas. A Yes/No question means that the input response is a choice made between Yes and No. A True/False question means that the input response is a choice between True and False. An N-Point question means that the question is a Likert scale with values with a range from 1-7. The meaning of those values is specified by the experimenter. In general, the Likert scale is used to measure the attitude that a person has towards a specific statement, idea, or question. The labels for the Likert scale have 1 being the most negative attitude and 7 being the most positive, although traditionally, Likert scales usually scale from 1-5.

The format of the test files can be seen in figure 3.4. A test file first has information about general aspects of the experiment, such as title, general instructions, number of blocks, etc. This information is followed by the block information, trial information, and question information. Block information signifies the beginning of a block of trials, so there will be both trial information and question information in between block information. This same structure holds true for trials, as there will be question information that is between trial information. *TestCreator* writes out the file and *TestTaker*, discussed in the next section, reads in the file and presents the trials to the test subjects.

3.4 TESTTAKER

TestTaker is the component of the VizEval Suite that is the focus of this thesis. The input to *TestTaker* is a test file that was created using *TestCreator*. The first screen in *TestTaker* is the test selection and information gathering screen. This screen is where the experimenter selects the test file that is to be used. After selecting the test file, the experimenter will specify the

directory that the log file should be written to. After specifying the directory, the user name (should be unique) is then input, followed by the user's distance from the screen, the screen width, and the screen resolution.

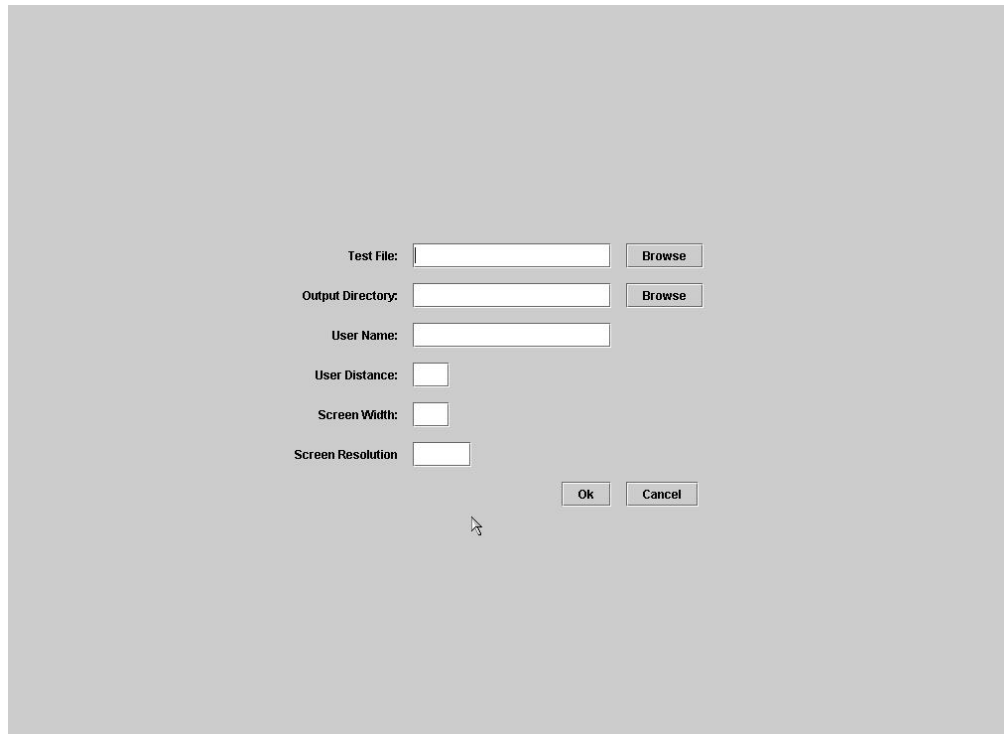
The image shows a software window titled "TestTaker Introductory Screen". It contains several input fields and buttons. The fields are labeled "Test File:", "Output Directory:", "User Name:", "User Distance:", "Screen Width:", and "Screen Resolution". The "Test File" and "Output Directory" fields have "Browse" buttons next to them. The "User Distance", "Screen Width", and "Screen Resolution" fields are smaller. At the bottom right, there are "Ok" and "Cancel" buttons. A mouse cursor is visible near the bottom center of the window.

Figure 3.5: *TestTaker* Introductory Screen

The next screen that would be encountered would be the experiment instruction screen. This is where the participant would read any instructions that would be necessary to give them pre-testing knowledge. The information could range from how long the test is going to be to specific instructions about certain questions. When the user clicks on the “next” button, they then proceed to the block instruction screen. This will contain more detailed instructions that pertain specifically to a block of trials. Having block instructions allows information to directly precede the series of trials that they pertain to.

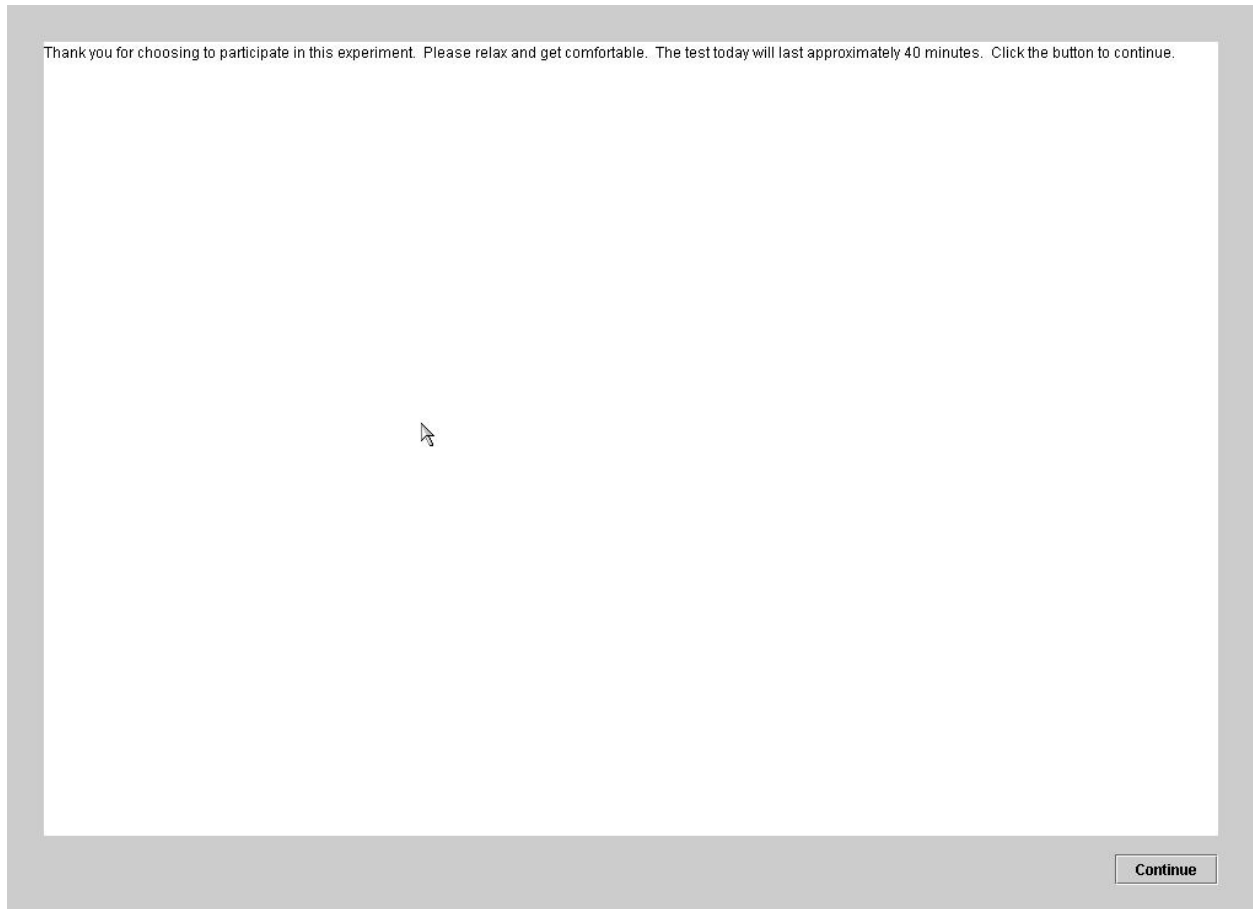


Figure 3.6: *TestTaker* Instruction Screens

After the block instruction screen, the actual interactive part of *TestTaker* is displayed. The test file has already been read to retrieve the instruction information for the test and for the first block. Now, the test file is read one trial at a time, and closed after each trial is read. The original design read in the entire test file. However, as the number of trials increases, the amount of memory used becomes wasteful and tends to slow down other processes. The current design permits on-the-fly changes to the experiment file. After execution and completion of one trial, the file is reopened and the next trial is read in.

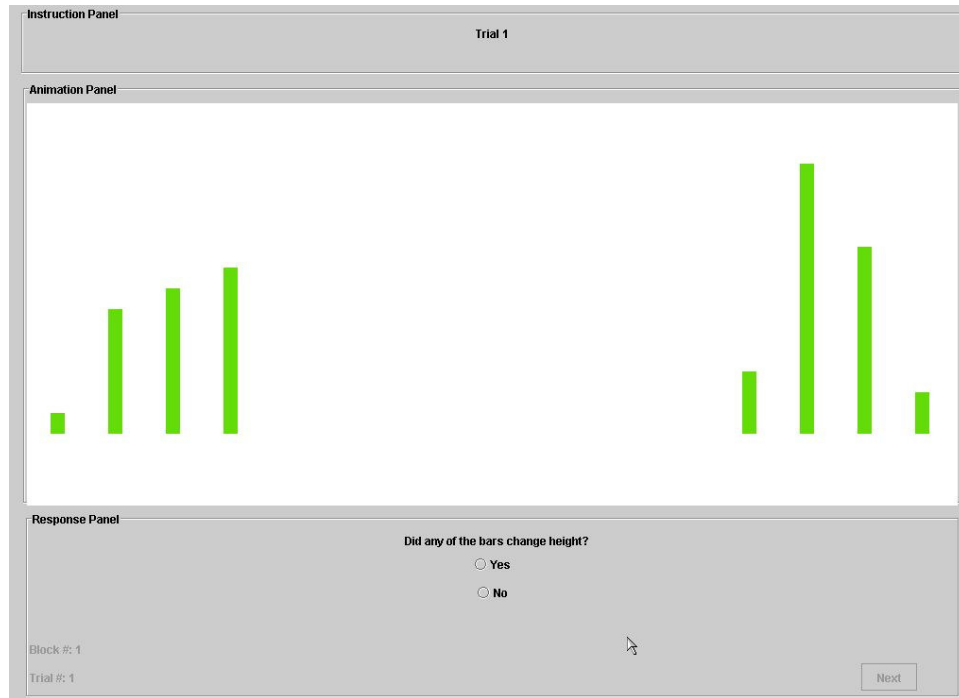


Figure 3.7: *TestTaker* Main Interface

At the end of the block of trials another screen is displayed. It is a screen that contains block ending instructions. This allows the experimenter to give instructions that would be helpful once a block of trials is complete. For example, if after the block of trials the participant should contact the experiment administrator before continuing, then that is what could be used as instructions here. *TestTaker* then proceeds to read in the next set of block start and block end instructions, followed by the first trial in that block.

It is important to note that all of the files that are used in the VizEval suite are text files. The decision to use regular text files was a well-considered design decision. We wanted the files to be platform independent, because every platform recognizes a text file. Second, we wanted the files to be simple and easy to read. Previous versions of the VizEval suite worked with files that could only be read with specific applications that were created just for them, such as *TestCreator* or *TestTaker*. This meant that it was impossible to edit or read the file without using the application. Having text files gives the freedom to be application independent as well. Now,

graphics, animation, and test files can be viewed and edited with a simple word processor such as Notepad. This allows modifications to be made very easily without running any applications. This gives the files maximum flexibility in their use. Also, the files can be modified at any time. So if a change needed to be made to any of the graphics or animation files as well as the test file, the experimenter would just open it up, make the necessary changes, and save it. If it is a graphic or animation file, then the changes will be implemented the next time that the graphics or animation file is called from a trial. The test file will only recognize changes that occur in later trials, not current or previous trials.

CHAPTER 4: EXPERIMENT

4.1 INTRODUCTION

Our first experiment, Experiment 1, was designed to test out the VizEval Suite software, and to collect some preliminary data for the study. Future experiments will be created and implemented in VizEval Suite to test other visualizations, cueing styles, etc.

4.2 SETUP

Experiment 1 evaluates the effect of cueing on perception of changes in the animations. The graphical objects are rectangles and the cue involves having a bar flash twice. After the cue, the bar may change by either decreasing or increasing in height. The bar heights that are selected range from 44 pixels to 374 pixels, at 22 pixel intervals. This allows for a maximum of 16 bars to be used. The reason for choosing the pixel value change of 22 is because that is the “just noticeable difference” in height change. The just noticeable difference in height change refers to the minimum value for a change that can be recognized when a focal point is being attended to. Pilot studies were conducted and the results concluded that the minimal change in height noticeable by the human eye while attending to a focal point is 22 pixels. The number of bars to be displayed would be 4, 8, or 16, with the 8 bars being a subset of the 16, and the 4 being a subset of the 8. The color used for the bars was green, with RGB values of 100, 220, and 10, respectively.

The animation files used are designed to test the effect of cueing on the viewers ability to recognize a change in bar heights. We cue 1 or 2 bars and change 0, 1, or 2 bars. The animation files used include those that: cue 1-change 0, cue 1-change 1, cue 1-change 2, cue 2-change 0,

cue 2-change 1, cue 2-change 2. As seen below in Figure 4.1, using a various combination of cues and changes allows the experimenter to test different things. By comparing the results from different tests, conclusions can be drawn about the attributes and/or cues used in the experiments. For example, in this experiment, these animation files are used in the experiment on the 4, 8, or 16 bar graphic files to cause them to change. The participants answer questions based on many combinations of graphic and animation files.

		Change		
		0	1	2
C u e	1			
	2			

Figure 4.1: Cue-Change Comparison

In the *TestTaker* application, a countdown timer is used to create a common focal point throughout the experiment. Before any animation can begin, the timer must countdown to zero. At this point, the zero remains visible throughout the entire length of the animation and disappears after the animation is complete. The participants are instructed to use the zero as a focal point during the animation. The bars are placed symmetrically from the edge of the canvas across the useful field of view. That means that in a four bar file, two bars are at the two farthest left positions and two bars are at the two farthest right positions. In an eight bar file, there are four bars that occupy the four farthest right and four farthest left positions. In a sixteen bar file, all of the available positions are occupied with a bar.

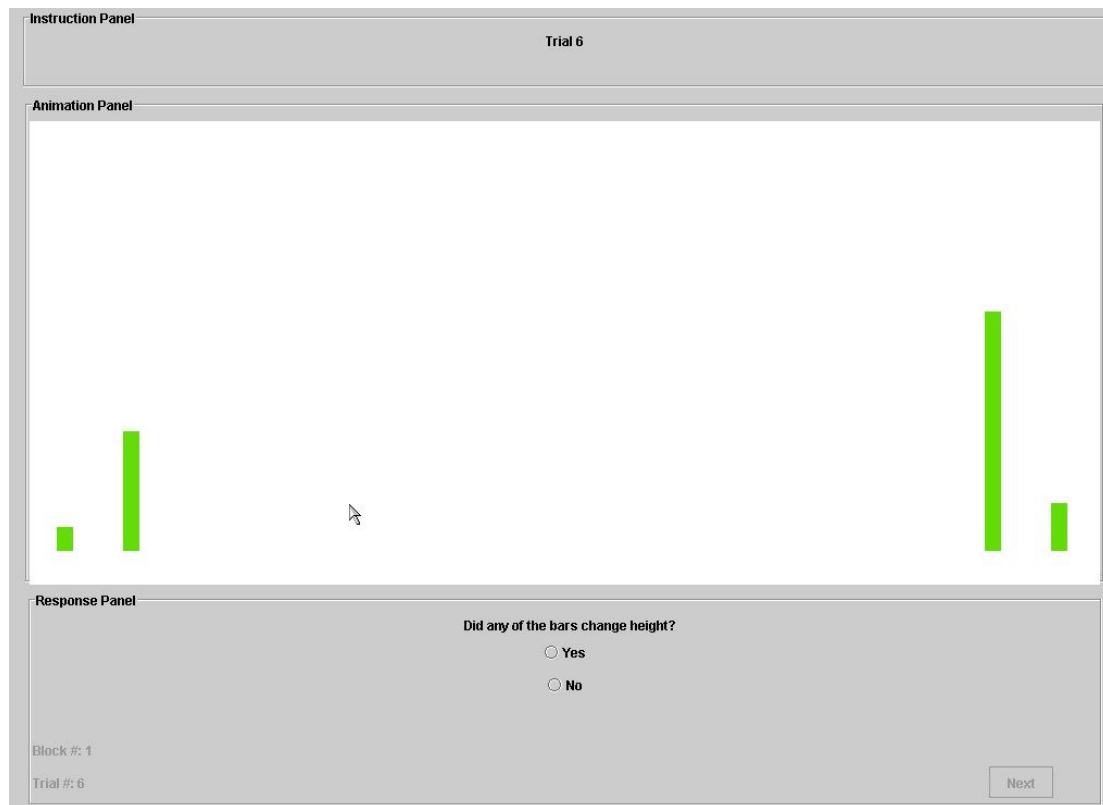


Figure 4.2: Sample 4 Bar Graphics File Placement

The tests were conducted on Dell Dimension 8300 PCs with 17" flat screen monitors. The participants were placed 50 cm from the screen while performing the experiment so that the display would be within the useful field of view. The first experiment was created to test all of the possible combinations of graphics and animation files, making 288 trials. Also, each trial contained 5 questions about the animation. The first question, a Yes/No question, was "Did any of the bars change height?" This was followed by a Click question that said "Please click on the bar most likely to have changed height.", followed by an N-point question "How confident are you that this bar actually changed height?", with 1 being "Not at all confident" and 7 being "Extremely Confident." The next question in the sequence was another Click question that said "Please click on the second bar most likely to have changed height." followed by another N-point question on confidence in the prior selection.

4.3 DISCUSSION

The collected data will be used to evaluate factors such as the effect of cueing on helping users to 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 that of future experiments to compare the results of various cueing strategies. A pilot experiment was run to test *TestTaker* and collect preliminary data. In the study, the participants completed 72 trials, a subset of the 288 trials. One result of the pilot study is that the participants correctly responded to 78% of the bars that changed height when they were cued compared to only 15% correctly responded to when the bars were not cued.

The resulting data shows that the majority of the participants responded more accurately to the bars that were cued prior to changing height. The result of the non-cue bars showed that the participants typically did not recognize changes in bars that were not cued. This preliminary analysis is based on a study of n participants. A larger study is underway that will permit an assessment of the statistical significance of this result.

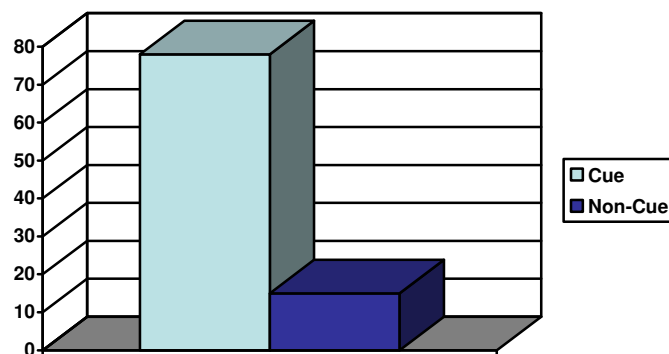


Figure 4.3: Table of Results for Experiment 1

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

From our experiment we conclude that tools such as *TestTaker* and the VizEval Suite permit researchers to conduct experiments on the effectiveness of program visualization. For example, our preliminary study found that cueing does have an effect on noticing whether or not a bar will change in height. The bars that were cued were more likely to be detected than those that were not cued.

5.2 FUTURE WORK

Currently, *TestTaker* does not support the keyboard question type. The keyboard question will record which key is pressed in response to a question. It will also allow navigation through future tests. In the future we plan to implement the keyboard question type. For the multiple choice question, there is only the ability to have 5 options, but not 4 or 3. Future versions of *TestTaker* will have the ability to display 3, 4 or 5 options for a multiple choice question. For the N-point question that involves the Likert scale, there is only the ability to have the scale go from 1 to 7, but no larger or smaller. Future versions of *TestTaker* will implement a Likert scale lower than 7, but any larger scale would complicate the display and cause unnecessary scroll bars which may affect performance during the test. Also, after each question the participant must click on a “next” button to traverse through the test, but feedback seems to say that this is time consuming and very redundant. The suggestion is that the test advances after the user answers a question; however, this has a slight drawback. The answer, once selected,

will be the only answer that will be allowed. The participant will not be allowed to make any changes to their answers. We will have to pursue the matter further to see if that has an effect on the accuracy of the results or not.

BIBLIOGRAPHY

- [1] Baecker, R. *Sorting out sorting: A Case Study of Software Visualization for Teaching Computer Science*. In *Software Visualization: Programming as a Multimedia Experience*. Cambridge, MA: MIT Press, 1998, pp. 369-381.
- [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 *Proceedings of Interact 2001*, Tokyo, Japan.
- [4] Byrne, M.D., Catrambone, R., and Stasko, J. T. *Do algorithm animations aid learning?* Technical Report GIT-GVU-96-18, Georgia Institute of Technology, August 1996
- [5] Davis, E.T., Kramer, P., and Graham, N. (1983). Uncertainty about spatial frequency, spatial position, and contrast visual patterns. *Perception and Psychophysics*, 33, 20-28.
- [6] Denning, P.J. *A Debate on Teaching Computer Science*. *Communications of the ACM*, v.32 n.12, pp. 1397-1414, December 1989.
- [7] Faraday, P. and Sutcliffe, A.. *An empirical study of attending and comprehending multimedia presentations*. In *Proceedings of the ACM conference on Multimedia*, Boston, USA, 1996
- [8] Faraday, P. and Sutcliffe, A.. *Designing Effective Multimedia Presentations*. *Proceedings CHI'97 ACM*, 1997, pp.272-278
- [9] Healy, C.G., St. Amant, R., and Elhaddad, M. (1999) *ViA: A Perceptual Visualization Assistant*. In *Proceedings 28th Applied Imagery Pattern Recognition Workshop*, pp. 1-11
- [10] Hamilton-Taylor, A.G. and Kraemer, E., *Designing an Algorithm Animation System to Support Instructional Tasks*. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning (IMEJ)*, October 2002.
- [11] Hamilton-Taylor, AG.. and Kraemer, E. *SKA: Supporting Algorithm and Data Structure Discussion*. *ACM SIGCSE Bulletin*, *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, February 2002, Volume 34, Issue 1, pp. 58-63.

- [12] Pashler, H. E. (1998). The Psychology of Attention. Cambridge, MA: The MIT Press 1998
- [13] Stasko, J. T. and Kraemer, E. *A Methodology for Building Application-Specific Visualizations of Parallel Programs*. Journal of Parallel and Distributed Computing 18(2), June 1993, pp. 258-264.
- [14] Ware, Colin. Information Visualization: Design for Perception. Academic Press (1999).
- [15] Z.W. Pylyshlyn and C.R. Sears. *Multiple Object Tracking and Attentional Processing*. Canadian Journal of Experimental Psychology. 2000, 54(1), 1-14.
- [16] <http://www.cc.gatech.edu/gvu/softviz/parviz/polka.html>
- [17] <http://www.cc.gatech.edu/gvu/softviz/algoanim/samba.html>
- [18] <http://www.pstnet.com/products/e-prime/>
- [19] National Science Foundation Award # 0308063. *Program Visualization: Using Perceptual and Cognitive Concepts to Quantify Quality, Support Instruction, and Improve Interactions*. June 15, 2003 – May 31, 2005. Principle Investigator: Eileen Kraemer