2010

Quantum Algorithm Animator

Lori Eileen Nicholson
Nova Southeastern University, lonichols@gmail.com

Quantum Algorithm Animator

by

Lori E. Nicholson
lnichols@nova.edu

A Final Dissertation Report
submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Science

Graduate School of Computer and Information Sciences
Nova Southeastern University

2010

We hereby certify that this dissertation, submitted by Lori E. Nicholson, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

_____ _____

Michael J. Laszlo, Ph.D. Date
Chairperson of Dissertation Committee

_____ _____

Maxine S. Cohen, Ph.D. Date
Dissertation Committee Member

_____ _____

Junping Sun, Ph.D. Date
Dissertation Committee Member

Approved:

_____ _____

Leo Irakliotis, Ph.D. Date
Dean, Graduate School of Computer and Information Sciences

Graduate School of Computer and Information Sciences
Nova Southeastern University

2010

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Quantum Algorithm Animator

by
Lori E. Nicholson
November 2010

The design and development of quantum algorithms present a challenge, especially for inexperienced computer science students. Despite the numerous common concepts with classical computer science, quantum computation is still considered a branch of theoretical physics not commonly used by computer scientists. Experimental research into the development of a quantum computer makes the use of quantum mechanics in organizing computation more attractive, however the physical realization of a working quantum computer may still be decades away.

This study introduces quantum computing to computer science students using a quantum algorithm animator called QuAL. QuAL's design uses features common to classical algorithm animators guided by an exploratory study but refined to animate the esoteric and interesting aspects of quantum algorithms.

In addition, this study investigates the potential for the animation of a quantum sorting algorithm to help novice computer science students understand the formidable concepts of quantum computing. The animations focus on the concepts required to understand enough about quantum algorithms to entice student interest and promote the integration of quantum computational concepts into computer science applications and curricula.

The experimental case study showed no significant improvement in student learning when using QuAL's initial prototype. Possible reasons include the animator's presentation of concepts and the study's pedagogical framework such as choice of algorithm (Wallace and Narayanan's sorting algorithm), design of pre- and post tests, and the study's small size (20 students) and brief duration (2 hours). Nonetheless, the animation system was well received by students. Future work includes enhancing this animation tool for illustrating elusive concepts in quantum computing.

Acknowledgements

I would like to express my deep gratitude and sincere appreciation to my advisor, Dr. Michael Laszlo, whose patience, guidance and support helped me complete this memorable project. I am also most grateful to Dr. Junping Sun and Dr. Maxine Cohen for their continual support, encouraging reviews and for sharing their time and expertise in serving as part of the committee for my dissertation. In addition, I would like to thank the other Computer Science professors at Nova Southeastern University for providing a wonderful educational experience during my Ph.D. coursework.

At Wayne State College, I would like to thank my department chair, Dr. Timothy Garvin, for his patience and support as well as Dr. Vaughn Benson and Dr. Bob McCue for standing by me when opposing forces attempted to prevail. I would like to thank my mentors and friends, Molly Curnyn and Patricia Arneson for their kind words and encouraging jabs.

I would like to thank my parents, Jim and Charlotte, and my sisters and brother, Kristi, Teri and Gene, for their considerate prodding. Now our conversations do not have to start out with that one common question! I would like to thank my wonderful friend Laura, for her steadfast support and many years of patiently listening to me.

Finally, I would like to thank my husband, Jerry, and my son Evan for their encouragement, playful banter, and constant support. I could not have accomplished this monumental task without their love and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Problem Statement and Goal

In the early 1970s, theorists researched the possibilities that a quantum computer could be faster than a conventional computer for solving intractable problems. Academic curiosity encouraged these observations without notice until Shor (1997) attracted attention to the field of quantum computing with the development of a polynomial time algorithm for factoring large numbers. Shor's discovery launched interest in quantum computing and encouraged research into the development of an operational quantum computer. Current quantum computer research is still theoretical and functionally impractical but in the next few decades, quantum computers are likely to move out of research labs and into practical applications. Bacon and Leung (2007), provide a summary of the broad range of experimental methodologies used by researchers to build quantum computer's calling this quest one of the greatest technological races of the 21st century.

Similar to their classical counterpart, quantum algorithms predate the actual development of a practical quantum computer. Quantum algorithm research has provided solutions for many computational problems and in some cases, provided more efficient solutions than classical algorithms. Grover's (1996) algorithm uses quantum concepts to search an unsorted database faster than its classical equivalent. Searching an unsorted database using a classical search algorithm requiring a linear search runs in $O(N)$ time to find the requested input in a database of N entries. Grover's algorithm provides a solution to the same problem in $O(\sqrt{N})$ time

(Brassard, 1997). Shor's (1997) algorithm, with its simplification of factoring, could cause issues for public-key cryptography systems or move that technology into a new realm for information security. These foundational algorithms provide solutions that are more efficient for many of today's practical computing applications like database searches, data mining procedures and prime number factoring by utilizing the enigmatic principles of quantum mechanics.

Quantum programming is different from classical program development. In classical programming, knowledge of the underlying computer architecture is not required. Abstraction insulates the programmer from machine architecture. In contrast, quantum programming requires an understanding of the implications of quantum mechanics to quantum computation. Classical algorithms are written to accept some value, or set of values, as input and produce some value, or set of values, as output. The typical classical algorithm is a sequence of computational steps that transition from one state to the next in either a deterministic or a probabilistic procedure. Quantum algorithms typically start with a particular state similar to classical algorithms but from there they transition into a superposition of many states. A sequence of quantum state transformations leads to a measurement of the system. The likelihood of the system being in one state or another or in both is what defines the power and usefulness of quantum computing.

Shor (2003) examined the question of why there were so few classes of quantum algorithms since early foundational quantum algorithms were developed. Shor provided several reasons for this situation, one being that quantum computers are functionally different from conventional computers. Techniques used for developing classical algorithms and classical intuitions for understanding the process of computation no longer work in the quantum environment. Quantum physics has been a developing science for several decades, yet computer science has only just

begun to incorporate the use of quantum mechanics into their field. Shor suggests that future computer scientists will build upon early foundational research to discover numerous new and significant quantum algorithmic techniques.

Current work on the development of a practical quantum computer, active research in the area of quantum algorithms, and the inherent difference between conventional and quantum computing concepts establish a need for computer science programs to integrate quantum concepts into their curricula. Although many institutions have added courses and content related to quantum computing, there is a paucity of introductory tools available to promote the understanding of quantum computing and assist students in learning about quantum algorithms.

Educators in computer science departments have used algorithm animation as an introductory tool to present visualizations and organized animations to students. Numerous animation systems have been developed to aid students in experimenting with and understanding classical algorithms. These systems provide a means of displaying events as they occur in various phases of the algorithm and show basic operations of the algorithm. By illustrating how objects change, presenting modifications to internal data structures and exposing hidden algorithm properties, these systems provide an educational tool for learning algorithmic concepts.

This research provided a solution to the lack of introductory quantum algorithm pedagogical tools, using data gathered from classical algorithm animator systems to create a quantum algorithm animator. Information about existing classical algorithm animators along with research into quantum concept learning was used to design this quantum algorithm animator. A case study followed involving twenty computer science students to test the animator's effectiveness in learning quantum computing concepts and acquiring aesthetic and interface

suggestions for future improvements or alternative design considerations.

The goal of this research was to create a quantum algorithm animator designed to provide an interactive learning environment for students. The quantum algorithm animator was developed and entitled QuAL (Quantum Algorithm Animator). QuAL's main objective was to provide visualizations of quantum computing concepts that may lead to a better understanding of quantum algorithms. This thesis investigates how animations can be an effective means of conveying the dynamic behavior of quantum algorithms by using QuAL in the case study.

This research was accomplished in three stages. First, an exploratory study gathered information about existing classical animation systems and compiled a listing of successful approaches. Algorithm animators have been used as pedagogical tools since the early 1980s and their effectiveness has been an active research topic (Stasko, Badre, & Lewis 1993; Kehoe & Stasko, 1996; Gurka, 1997; Hansen, Schrimpsher, & Narayanan, 1998; Bryne, Catrambone, & Stasko, 1999; Ciesielski & McDonald, 2001; Vrachnos & Jimoyiannis, 2008; Urquiza-Fuentes & Velazquez-Iturbide, 2009). The results from these various research projects have concluded that algorithm animators can provide positive learning experiences.

After gathering the data from the exploratory study, this research utilized successful components from classical animators to determine initial design criteria for QuAL. Classical animators have taken numerous approaches to visualizing the internal operations of algorithms and this research segment focused on those techniques that provide appropriate visualizations for quantum computing concepts. Design criteria, user interface issues, and the creation process for the production of quantum algorithms were refined by this research to provide an effective learning environment. Techniques such as scaffolding, animation chunking, multiple levels of granularity, interactivity, and promoting critical thinking have been integrated into many of the

existing classical animator's frameworks. Did they support quantum algorithm learning and animate the concepts of quantum computing? Certain design criteria were found to be effective but inefficient when dealing with quantum computing concepts. Allowing the manipulation of input values was restricted during this initial prototype phase due to exponential growth when qubit registers were put into a superposition of states. The number of steps to complete a solution was redefined but the addition of code highlighting was added along with the ability to control the speed of the animation. Additional findings are described in greater detail in Chapter 4 of this disseration report.

Finally, this stage of the research investigated the architectures used to implement algorithm animation systems. Which architecture provided a rich, interactive learning environment for QuAL? Diehl (2007) examines the following architectures that have been used to create many of the existing classical algorithm animation systems:

1. Ad hoc: A single algorithm is animated and implemented from scratch.
2. Libraries: Single algorithms are implemented using existing libraries with built-in graphical abstractions, control elements, etc.
3. Special datatypes: Datatypes with built-in visualizations are used to program algorithms.
4. Postmortem: Separate applications are used for the algorithm and visualization tool.
5. Interesting events: Annotations are used at essential program points in the algorithm to display interesting events.
6. Declarative: Similar to interesting events except that the annotations and algorithm code are separated.
7. Semantics-directed: A visual interpreter or debugger is used to execute the

algorithm.

Gloor (1997) provided a ‘recipe’ for designing an algorithm animator. He proposes a seven-step process for taking pseudocode of an algorithm to animation. Gloor’s process begins at stage one with the creation of a script. Typically this is the algorithm’s pseudocode describing the high-level action occuring in the animation. Stage two builds a storyboard, which is a series of visualizations that illustrate the previous script or pseudocode. Next, in the layout stage, the data structures to be animated are graphically specified. If sound is used it must be incorporated into stage four which occurs directly before the final animation is created. Animation follows in stage five of this seven-step process: computer animation packages may be used, or scripting languages, or existing animation system frameworks. The final two stages consist of inbetweening and editing. Inbetweening is a technique used to smooth the transitions between keyframes and editing produces the finalized algorithm animation system.

Experimenting with the data discovered in the second stage of this research provided solutions for QuAL’s design. Quantum computing concepts such as amplitude, state transitions, probability distribution, qubit registers, entanglement, and unitary operations, were integrated into QuAL to provide the student with the knowledge they will need to understand quantum algorithms. QuAL will only animate one sorting quantum algorithm in this research’s initial prototype and case study.

The final stage of this research presents a framework for using QuAL as a pedagogial tool and a case study was performed experimenting with QuAL in an educational setting. QuAL is conceived as a tool to help students form an effective bridge between textual material and the application of quantum theory commonly used to develop quantum algorithms. The results of the case study discussed in Chapter 4 provided preliminary conclusions to QuAL’s effectiveness

as a pedagogical tool and information for future modifications to its functionality, interface, and design. Chapter 3 contains an outline of the case study preformed by this research.

Relevance and Significance

Algorithms are considered the cornerstone of computing (Levitin, 1999). Their concepts have been introduced in many computer science programs as well-defined computational procedures that may be considered tools for solving computational problems. Student understanding of algorithmic concepts is important to the field of computer science but learning to develop and understand algorithms may be difficult because the algorithm is a static description of a dynamic process.

Algorithm learning has been researched for many years as educators pursue a methodology that might help students learn these concepts more efficiently and completely. Animating algorithms is one method that has been used to visualize their internal operations and display concepts using graphical representations. Animations were the logical next step to illustrations of data structures and algorithm workings found in earlier computer science textbooks and they provided a means of manipulating or controlling the display.

Animating algorithms for educational purposes has motivated the development of animation systems starting in the early 1970s (Hopgood, 1974). Baecker (1981) was the first to introduce video as a medium for illustrating a number of different sorting algorithms running on both small and large datasets. His 30 minute video entitled “Sorting Out Sorting” was a color film that added a new dimension to teaching algorithms and was the first to use sound. The video worked so well that students were able to watch the animations and successfully program some of the algorithms described without any further instructional information. These animations were viewed by students of various levels of computer expertise in this case study, but were mostly

used in introductory courses on algorithms and problem complexity (Stasko, Dominque, Brown, & Price, 1997).

Building on this foundational research, Brown and Sedgewick (1984) at Brown University developed BALSA-I which was the first interactive algorithm animation system supporting multiple simultaneous views of an algorithm's data structures. Stasko (1990) followed with the development of TANGO, an animation system that introduced the path-transition model. Many other systems began to emerge providing different approaches to animating algorithms like BALSA-II (Brown, 1988), CAT (Brown & Najork, 1993) and more current systems such as CATAI (Cattaneo, Ferraro, Italiano & Scarano, 2002), ANIMAL (Robling, Schuer & Freisleben, 2000), and LEONARDO (Crescenzi, Demtrescu, Finocchi & Petreschi, 1997).

Subsequent research following the development of animation systems focused on providing evidence of the benefits of using algorithm animation as an educational tool (Byrne, Catrambone & Stasko, 1999; Hansen, Schrimpsher & Narayanan, 1998; Kehoe, Stasko & Taylor, 2001). Mixed pedagogical results were found but the majority recorded slightly better results in situations using visual animations. According to Tudoreanu, Wu, Hamilton-Taylor and Kraemer (2002), using animation as a learning tool does provide benefits for understanding algorithmic computations but they speculate that the animation system should avoid unrelated activity and focus on using graphics that are self-explanatory and allow the user to connect directly to the computation. In addition to teaching algorithmic concepts, animations have relevance in other areas of computer science. Jones and Newman (1996), utilized animation techniques as an instructional tool to teach operating system concepts, and Leung (2005), developed a visualization tool for learners to gain insight on the Linux scheduling algorithm.

Diehl (2007) describes several scenarios that have been developed to achieve higher learning

involvement particular to algorithm animation. The first explores the functional structure supporting exploratory learning by only providing the primary building blocks of the algorithm. In using this type of an application the user actively reinvents fragments of an algorithm providing the student with an educational experience that finds the steps of a function for a specific data input. The next scenario consists of visualized path testing that focuses the user on finding input data for an algorithm that satisfies precise criteria rather than reconstructing the algorithm. This type of an application would provide certain visualization tools allowing for specified criteria and performance tasks.

Research in human-computer interaction (HCI) has lead to the emergence of a field of study called information visualization. This field has added interesting tools and techniques for the development of interfaces with special attention focused on the relationship between the user and visual representations of abstract data. Shneiderman (1973) pioneered visualization research techniques as a path to concept discovery when he developed a polynomial viewer that used an interactive visualization interface. Continuing his research, Shneiderman (1983) introduced a concept called direct manipulation which provided an interface style involving the continuous representation of objects of interest with fast, reversible, incremental actions and feedback. Many of Shneiderman's visualization methods are still used as powerful components visualizing technology in professional and educational settings (Plaisant & North, 2007).

Data gathered from studies in this area have provided principles and guidelines for improving the development of high quality interface designs. Shneiderman and Plaisant (1998) present the future of interfaces with regard to higher resolution screens and the web as well as provide insight into redesign issues with workplace software using ethnographic studies (Rose, Shneiderman & Plaisant, 1995). Some of the tools that have been developed based on early

information visualization research have provided user-controlled applications making the creation of dynamic visualizations quick and effective (Plaisant & Vinit, 1994; Heer, Card & Landay, 2005).

Shneiderman, Plaisant, Cohen and Jacobs (2009) continue HCI research by discussing dramatic changes in user-interface design that can assist the learner in visualizing and interacting with different concepts. Specifically for information visualization, their research claims that techniques can be used to provide graphical presentations and user interfaces for manipulating data with a larger number of items. Quantum mechanical concepts can become complicated by the fact that many mathematical manipulations utilize matrix multiplication that may produce a large amount of information. Visualization of this data may have to rely on Shneiderman et al.'s techniques. Their research will also provide useful HCI techniques for analyzing and interacting with QuAL.

Scientific discovery learning can be a highly self-directed process and students may learn more by using familiar concepts as building blocks to understanding new or unfamiliar concepts. Sorting data has historically been a fundamental problem discussed in many introductory computer science curriculums. Sorting algorithm research has added a significant number of solutions to this problem which has created a broad range of learning tools for this topic. This research used sorting concepts as a means of bridging the gap between classical and quantum algorithm understanding. The main purpose for using sorting concepts in this research is not to learn about sorting algorithms as students using QuAL should already have acquired this information from another source. The typical user of QuAL will be a first or second year computer science major with a basic understanding of the foundational principles of classical algorithms. Therefore sorting concepts will be used to cover the principles of algorithm design

and analysis using an extremely intuitive problem (i.e. sorting). Visualizations of sorting algorithms have been a very popular research topic. Grissom et al. (2003) compared levels of student learning engagement in their research by using the sorting problem as a simple learning algorithm. Grissom's group concluded that visualizations of the sorting algorithm did indeed assist students in learning about algorithmic procedures. In another article Rasala, Proulx and Fell (1994), used sorting algorithms to provide students with a comparative approach to algorithm analysis. Rasala et al. decided to use the sorting problem in their research because students learn simple sorting concepts early in their course work, sorting concepts are easy to visualize as they are familiar to students, and there are many sorting algorithms. With many different sorting algorithms, animations can provide comparisions by illustrating more efficient techniques, benchmarking different solutions, and running side-by-side simulations.

Quantum sorting algorithms have been a topic covered by past and present research with a common finding that many classical sorting algorithms are just as efficient as quantum algorithms. Klauck (2003) found this to be only when the analysis was based on the number of comparisons alone. Hoyer, Neerbek and Shi (2002), provided evidence that no comparison-based quantum sorting algorithm could outperform a classical sorting algorithm. Yet, in space bounded sorting, Klauck (2003) found that quantum computers outperform conventional computers significantly and when lower bounds techniques are used to solve quantum sorting problems, quantum search and sorting algorithms provide more efficient solutions than classical algorithms (Yao, 1994; Ambainis, 2000; Buhrman & de Wolf, 2002). The efficiency of quantum sorting algorithms will not affect the outcome of this research as the reason for using the sorting problem is not based on comparison benchmarking rather on the common foundational knowledge that it provides between classical and quantum algorithmic concepts.

Current quantum computing research focuses on varying approaches towards developing a practical quantum computer, the study of quantum information, and optimizing established as well as developing new quantum algorithms. Bennink (2008) discusses the field of quantum information and presents concerns on how secure encryption methods would be if a quantum computer were developed but at the same time claims that quantum communication would provide significant advances in information security. From a computer scientist's perspective, Jorrand (2007) provides insights into the quantum information processing concepts that should interest academia and industry. Jorrand claims that with current breakthroughs in quantum concepts, the transition from classical to quantum has great potential for future research especially for the computer science field. Quantum algorithm research is also actively pursuing new and innovative solutions. Saeedi, Zamani and Sedighi (2008) have proposed a new algorithm entitled MOSAIC, which they postulate is more efficient, producing results in fewer steps when compared to recent search-based synthesis methods. Designing quantum algorithms can be a difficult process but Aharonov and Ta-Shma (2003) have introduced a new approach they define as Adiabatic State Generations (ASG). This new approach may bring new insights and methods into quantum algorithm development.

These articles provide just a few examples of current quantum computing and information research. A National Science Foundation report stated "NSF should fund continual research into understanding the ultimate algorithmic power of quantum computing along three fronts: Developing new algorithms, understanding and developing new techniques used by quantum algorithms and optimizing and extending current algorithms" (Theory of quantum computing and communication, 2002, p. 3). This statement along with the above research provide evidence that quantum algorithm development is an active research topic, therefore computer science

curriculums should add these concepts into their current coursework to promote student interest in quantum computing.

Computer science continues to evolve as new innovative ideas emerge from scientific research. Conceptual shifts have occurred throughout computer science history. Programmers changing from structural to object oriented programming or single to multi-threading have added fuel to this evolution. Academia has responded by developing and using tools to assist students in learning these new concepts as the scientific community provides evidence that knowledge in these areas advances the field. Quantum computing is one of those areas that may alter the approach used by computer scientists to develop new and efficient algorithms. It has united the disciplines of physics and mathematics; understanding its concepts is becoming an important addition to expanding computer science student's horizons.

Barriers and Issues

Research in quantum computation is theoretical and there is no practical implementation of a quantum computer to date. Consequently, the amount of literature supporting the research and development of a quantum algorithm animator is limited. A few introductory teaching tools that animate certain aspects of quantum algorithms and several animation systems that incorporate quantum computing concepts exist but animating quantum algorithms has room to grow.

Emberson (2002) conceived the idea of using a quantum algorithm designer as a pedagogical tool for understanding quantum computer concepts in his Master's thesis. He entitled his application Quantum Algorithm Design (QAD) and developed it to allow users the ability to design and simulate quantum algorithms using quantum network gates with the circuit model of quantum computation. The user would be required to understand certain quantum mechanical principles before being able to design their first quantum algorithm. Hogg (2000) also

demonstrated the use of animation with a tool that used Mathematica as an interesting environment for animating Grover's search algorithms and comparing it to other quantum search algorithms.

Emberson and Hogg's research rely on special environments, software, and toolsets for their animations to function and assume knowledge of quantum physics. Knowledge of these environments or acquisition of these applications makes their animations less portable and perhaps costly to utilization in a classroom situation. Both designs also require precursory knowledge of hefty quantum mechanical concepts requiring the need for additional physics classes or curriculum adjustments. One of the goals for this research project was to identify the best choice for developing an animation system that provided ease of use, portability for educational effectiveness, and the appropriate development environment for introductory quantum concept visualizations.

One of the challenges that this research had to overcome was to depict the quantum computational process in a way that elucidates that process. Many resources currently available to the student assume or provide knowledge of quantum mechanics at levels that may deter a beginning learner away from these concepts. Quantum mechanical concepts are important in the development of quantum algorithms and this research had to balance the addition of certain quantum mechanical and quantum computing concepts with the presentation of material in order to encourage student curiosity. This balancing act was achieved by providing only what is needed for a basic understanding of quantum computing as the foundation for introducing quantum algorithms. Two objectives that helped to support this design goal:

1. Provide students with an introduction to basic quantum computing concepts i.e. entanglement, quantum state, interference, amplitude, and probability distribution.

2. Give students a clear understanding of the steps required by a quantum algorithm to solve a specific problem.

The development environment for this animation system utilized quantum computing concepts in a conventional computer environment in order to animate the quantum algorithm selected by this research. Quantum mechanics states that until they are observed, particles exist only as a discontinuous probability function. This situation is often visualized by conceptualizing the state of an unobserved particle as an overlay of all its possible states simultaneously. As an example consider a particle that might be observed in state X, Y, or Z. When it is not being observed, quantum mechanical principles claim that this particle may exist in three states simultaneously. The unobserved particle is said to be in a superposition of states. Current quantum computer simulators create an environment where particle superposition provides a reliable format in which an individual bit is stored as some measurable property of a qubit and can store bits that are simultaneously 1 and 0, unlike conventional computers where at any given time, a bit is either 1 or 0 but not both. These simulators can construct quantum logic gates based on the interactions of one or more qubits and depending on such processes as interference or entanglement, can be used to perform logical or mathematical operations on qubits executed in parallel. This research incorporated a quantum simulator into the development of this animator as a means of providing qubit registers, quantum data structures and pre-developed quantum functions. Quantum Computing Language (QCL) was used to provide this environment and is easily installed on a Linux system. Developing the animator as a Java applet provided easy interaction with QCL and did not reduce the importance of portability. Chapter 3 contains more information on the specific functionality of QCL (Omer, 1998).

Delimitations and Limitations of the Study

Delimitations of the study include:

1. The case study employed twenty computer science students from Wayne State College located in Wayne, Nebraska. The uniqueness of the study within a specific context makes it difficult to replicate the use of the prototype or travel to another location in another context (Creswell, 2003).
2. The study gathered only the students' perspective of the quantum animator's effectiveness through their personal experiences when using the animator.
3. The quantum animator was developed for educational purposes as an initial prototype not for professional developer purposes.
4. The quantum animator prototype developed concentrated animating one sorting quantum algorithm. Future developments may include other foundational as well as different classes of algorithms.

Limitations of the study include:

1. Because of the convenience sampling used, the researcher cannot say with confidence the sample used during the case study phase, will be representative of the population (Creswell, 2002).
2. Due to the nature of qualitative research, the data obtained during the case study phase may be subject to different interpretations by different readers.

Definition of Terms

Amplitude: Complex numbers used to express the quantum system (Yanofsky & Mannucci, 2008).

Animation chunking: Animations are broken up into meaningful smaller sized components (Diehl, 2007).

Decoherence: The quantum phenomenon that can cause random errors in a quantum system as a result of entanglement with the environment (Yanofsky & Mannucci, 2008).

Distributed Algorithms: Algorithms that run concurrently on many interconnected processing elements called processors (Koldehofe, Papatriantafilou, & Tsigas, 2006).

Interference: In wave theory, it is the effect of producing a new wave pattern when two or more waves collide and combine. This same effect can occur in the quantum world when amplitudes instead of heights interfere. Amplitudes are complex numbers and when interference occurs, they can add up in ways that cancel out (Yanofsky & Mannucci, 2008).

Hadamard Gate: A special quantum gate representing the building blocks of quantum circuits represented by the Hadamard matrix which is a unitary matrix (Omer, 1998).

Permutation: A rearrangement of elements in an ordered list into a one-to-one correspondence with itself. The number of permutations on a set of n elements is given by n! (Dickau, 2010).

Superposition: When a quantum system is in more than one quantum state at a time it is said to be in a superposition of states. Once the system is observed, the superposition will collapse into a measured state (Yanofsky & Mannucci, 2008).

Scaffolding (animation): Different techniques used to provide instructions for using interfaces or features. They may take many forms such as; guided tours, maps or overview diagrams, tables of contents, dynamic overviews (Diehl, 2007).

Transition State: In quantum mechanics, the state in which a quantum machine cannot be observed or measured (Yanofsky & Mannucci, 2008).

Quantum Entanglement: In quantum theory, when two or more object's interact with each other in such a way that their two quantum state's become entangled or act as if they were one single state. This entangled state results in a connection between the two objects, no matter how far apart they are taken (Yanofsky & Mannucci, 2008).

Quantum State: In quantum computation, is any possible state in which a quantum computer can be. They are described by state vectors and can be a 1, 0 or a probability of all possible states (Yanofsky & Mannucci, 2008).

Qubit: A qubit or quantum bit is the smallest unit of information describing a two-dimensional quantum system. Their classical counterpart is the bit, which describes a two-dimensional classical system (Yanofsky & Mannucci, 2008).

Summary

The purpose of this study was to examine the use of animation as a teaching tool for quantum algorithm behavior and develop a prototype for use in an educational setting. There has been limited work in this area with most of the work focusing on a specific development environment or requiring advanced knowledge in quantum principles.

Quantum mechanics has been an emerging discipline for many fields for some time but the computer science field is just beginning to embrace its properties. Several of these properties may provide efficient solutions to existing problems as well as provide the tools needed to promote new discoveries that may lead to faster database searches, better security protocols, and other interesting technological advancements.

Algorithm animators have provided visual representations for classical algorithms and have

been used by academia as a successful tool for teaching their concepts. The complexity of quantum algorithms may inhibit early computer science students from understanding the abstract behaviors thereby deterring interest in this field of study. Animation may reduce student fears by visualizing quantum behavior that introduces them to a different problem-solving paradigm.

This research gathered data from existing classical animation systems and incorporated them into a quantum animator called QuAL using principles that provide visualizations specific to quantum concepts. A case study methodology was utilized to assess the proposed educational framework from a student's perspective.

Chapter 2

Review of Relevant Literature

Classical Algorithm Animation Research

Animating algorithms has been an accepted method for teaching algorithmic concepts to computer science students despite mixed pedagogical successes. Most researchers have agreed that the benefits of past, present and future animation systems are dependent on the circumstances of the learning environment. Stasko (1997) claimed that students are enthusiastic about using visualizations, but studies have not provided enough evidence that animations have significantly improved their ability to learn certain concepts (Badre et al., 1991; Bryne, Catrambone, & Stasko, 1999; Stasko, Badre, & Lewis, 1993).

Although this early research did not provide clear results as to whether animations were effective pedagogical tools for algorithm learning, other research attempted to focus on finding out how animations could become effective pedagogical tools. Hundhausen et al. (2002) presumes that the way students use visualizations is more important than the animations systems themselves. Hansen, Schrimpsher, and Narayanan (1998) conclude that animations are an effective teaching medium but a rethinking of algorithm animation design is required. They continue by presenting a framework called Hypermedia Algorithm Visualization (HalVis) that utilizes multiple media, semantic links and numerous cognitive devices, which help students visualize algorithms specifically sorting algorithms. Hansen et al.'s conclusions provide evidence that designing an animation system to support incremental learning with real world analogies, which transition to algorithm concepts, assist students more than just providing some

textual feedback. Access to fundamental algorithmic building blocks and a learning objective-based design approach allowed their design to divide dynamic information into more manageable pieces that uses animation chunks. Features such as pause / repeat procedures, animation chunks presented in synchrony, three animation views (analogical, micro-level and macro-level), the ability to change inputs, performance predictions, and pop-up reflections, all contribute to a better learning environment.

Urquiza-Fuentes and Velazquez-Iturbide (2009) furnish readers with a comprehensive survey of successful algorithm animation systems along with main features that contribute to their success. The researchers added specifics on the educational effectiveness of certain features based on the conclusion that educational improvements could also depend on system-specific features. Urquiza-Fuentes et al. classified their evaluations by two criteria: abstraction level and implementation approach. Price et al.'s (1998) software visualization taxonomy defined the criterion of the abstraction level as: algorithm visualizations and program visualizations. The implementation criterion were defined as three systems: script-based, interface, and compiler-based. This study found that most animations systems contained two common properties: first, all systems were interactive and second that all focused on usability. These were important concepts for Urquiza-Fuentes et al. especially since they were evaluating these systems for educational purposes. Urquiza-Fuentes et al.'s recommendations for the design of algorithm animations based learning experiences are as follows:

1. Viewing animations provides more effective knowledge acquisition.
2. Include additional text and narrative comments.
3. Explicit feedback is very important when students are answering questions.
4. Provide the ability to change inputs with advanced features such as different execution

scenarios, integration with an IDE, and a manipulation interface .

Conclusions of this study find that script-based systems are more suitable for viewing and responding levels and compiler-based systems are more effective for changing, construction, and presenting levels.

Wiggins (1998) performed an interesting comparative study between two algorithm animation tools. Wiggins compared a textual based tool with a more graphical based tool both containing a graphical user interface and looked for a difference between student understanding using both the tools and a difference in student preference for either tool. The assessments used in this study were short answer exams, free response exams, and a preference questionnaire. Conclusions from data collected in this study found that there was no significant difference in student understanding but students tended to prefer the textual based system over the more graphical based system.

Historically, algorithm animation began by providing static illusions demonstrating the algorithm's concepts. The early versions of these pictorial representations could be manipulated to provide a high level of abstraction hiding details not relevant to the current concept. This process helped students step through certain details of the algorithm before adding more advanced items that might confuse them. These static visual representations became the precursor to dynamically animating algorithms which were represented, at first, as a sequence of illustrations (Gurka, 1997).

Once algorithm animation systems were introduced, research moved towards developing new more effective systems varying in their capabilities, platforms, support, animation libraries, target audiences, and user interfaces. Algorithm animation systems were also built for specific algorithm domains. Ayellet (1994) developed an animation system for use with geometric

algorithms. Geometric algorithms are used for solving geometric problems and are important for computer graphics, robotics, and pattern recognition applications (Dobkin, 1992; Latombe, 1991; Schwartz & Yap, 1987; Toussaint, 1986). Ayellet's system called GASP demonstrates the techniques used to present complex ideas in an aesthetic format for user appreciation of complicated mathematical notions. GASP animates highly complex geometric algorithms easily without precursory knowledge of computer graphics and is oriented towards the knowledge of the users. GASP's conceptual model includes three tasks: implementation of the algorithm annotated with interesting events to be animated, design and implementation of the animation, and interactive exploration of the algorithm. GASP also contains four components: the animation system, algorithm implementation, hooks to the system, and style files. An important concept that GASP demonstrates is that picking and well-defining a small domain makes it easier to create an animation system that enables ease of use and effective visualizations.

Jackson and Fovargue (1997) used the XTANGO software system to animate genetic algorithms (Stasko, 1990). Genetic algorithms are used to to find solutions to combinatorially hard problems. The difficulty in understanding genetic algorithms comes from the fact that their behavior is difficult to predict as their operations are driven by evolutionary principles as they attempt to solve highly mathematical problems. Jackson et al. developed a set of animation sequences specific for genetic algorithms used with lecture based materials to teach students about these interesting concepts. Their system is based on the assumption that students will be introduced to evolutionary concepts before using the animations. An important component of their teaching framework is the fact that they attempt to use an application that is simple enough to animate but realistic enough to capture the student's interest. Pedagogical considerations assisted with the design of the algorithm animations, the algorithm should be simple keeping just

the concepts needed to provide an introductory understanding of genetic algorithms as well as keep the genetic concepts simple. They excluded advanced genetic concepts in an attempt to softly introduce these difficult behaviors to the student and simply promote algorithmic understanding and interest. Jackson et al. didn't feel that execution speed or optimal solutions were important but focused instead on demonstrating significant genetic algorithm properties with trends and acceptable solution visualizations. The dynamic animation used by their system provided chromosomal visualization simulating real-world genetic properties as well as the addition of color to highlight their problem principles. User input provided the ability to change parameters such as number of chromosomes and genes, the probability of mutation and other important genetic algorithm parameters.

Tudoreanu, Wu, Hamilton-Taylor, and Kraemer (2002) provide evidence that algorithm animation promote the understanding of distributed algorithms in their research using 3-D visualization and legends. The methodology employed to assess student learning allowed users to view the visualizations while answering questions supplied by this study, which helped to promote their goal of testing effectiveness of the visualizations instead of the testing environment. The legends designed for their visualizations provide the student with a pictorial bridge between the encoded algorithm properties and the graphical features they represent. This allowed the system to be presented to the student without any explanation except for what the algorithm does. The classical termination detection algorithm for distributed systems was used in this empirical study to provide a distributed computation problem. Prior to testing, the students were shown a powerpoint presentation that explained the experimental environment and a textual document containing a description of the algorithm. The data collected by this study provided some evidence of the benefits to using visualizations in distributed algorithm

animation. Students did gain a more accurate understanding of the algorithmic concepts used in this study with the research results recommending the importance of reducing cognitive load to improve effectiveness.

Another study using distributed algorithms was completed by Koldehofe, Papatriantafilou, and Tsigas (2006) who entitled the environment LYDIAN. This environment provided support for teaching and learning a collection of distributed algorithms used in an undergraduate computer science class. LYDIAN provided students with the ability to write their own algorithms or select from one stored in a database allowing teachers to use LYDIAN in numerous formats with different levels of student expertise. The user interface was developed using TCL/TK with protocol-defined events written into a trace file. The user can choose between graphical or textual output and view all relevant information in one window. The objectives of this development were to provide the student with key concepts concerning distributed algorithms, change timing and workload to demonstrate different behaviors, and display communication and time complexities of the selected algorithm. Stasko (1995) developed animation libraries called POLKA that were used for this study and claimed to provide portability, friendly interface, and good visualization features like multiple views, speed tuning, step-by-step execution, and callback events.

Animation techniques as they relate to the art of dynamic algorithm visualizations have been a popular research topic as each new study attempts to provide more insight into the design of more effective dynamic graphics. New techniques versus modified older techniques have been manipulated and enhanced in an attempt to find the best mix for an effective animation system. Early in classical algorithm animation history, Brown and Hershberger (1991) demonstrate the importance of the use of sensory tactics by describing techniques that focus on the addition of

color and sound. Early versions of animators would use textual based displays falling into one of two categories. Monolithic view, which concentrates all of the algorithm's concepts into one dynamic view. These displays were successful for simple algorithms but would fail when attempting to portray complicated processes due to the fact that they had to encode so much information that the user would become confused or quickly tire of its details. And the other category, multiple views, which became the technique of choice as they allowed the user to see limited information about a few aspects of the algorithm. These views were easy to comprehend with the composition of several views more manageable and understandable than the sum of their individual views. With this in mind they provide summaries of several different systems and the techniques used to promote more effective visualizations. Zeus used a multilevel adaptive hashing technique that provided menus for the selection of algorithm, views, and input data (Brown & Hershberger, 1991). The control panel format provided multiviews with snapshot and restore capabilities, start/stop/step-thru buttons, as well as a slider for execution speed control. Color was added to several speciality views to provide differentiation of important concepts. Sound effects were also added to provide concept specific tones when elements were changed or simulated collisions occurred.

Sangwan (1997) built upon Brown and Hershberger's research by developing an animation system using self-visualizing C. This research highlights animation techniques such as:

1. Multiple views: several views providing significant aspects of the algorithm
2. State cues: the display contains changes in the state of the algorithm portrayed by changes in their graphical representation typically using a change of color or shape.
3. Static history: static material similar to the approach used by textbooks to convey the dynamic behavior of the algorithm.

4. Input data: data in the form of small input data, pathological data, or cooked data that helps the user step through concepts with visual displays.
5. Continuous versus discrete: small data sets support a continuous change while larger data sets favor discrete change.
6. Multiple algorithms: comparisons of several algorithms synchronously help the user compare and contrast specific concepts.
7. Color: the use of many united multiple views by representing similar or related objects or features. May also be used to highlight areas of interest, capture history, or reveal the state of an algorithm.
8. Sound: used to enhance visual views or signal omissions.
9. 3D: may be used to with objects or concepts that are inherently 2D.
10. Fisheye views: used to display large information structures.

The use of self-visualizing C in this research enables the animations to be written in C and the language itself adds a self-animating data type *int* that corresponds to C's *int* data type. The benefits of using this language provided ease of algorithm development and animations by both the researcher and the users of the animation system.

Current animation systems have used differing tactics to continue building more efficient visualizations and also focus more on the student learning environment. Chen and Sobh (2001) combined the visualization of data structures with the ability for user-defined algorithm animations in their software application for use by their introductory computer science students. Their software contained the following observable data structures: array, stack, queue, binary search tree, heap and graph. The user-defined component of their applicaton allowed creation of algorithms using the JavaMy language, which was used to visualize the execution of the

algorithm as well. The bubble sort algorithm was used to demonstrate the JavaMy code. An example of the translated bubble sort algorithm into JavaMy code (refer to Figure 1).

```
public static void main(String[] args) {
final int SIZE = 8;
MyArray intArray = new MyArray(
    AnimatorFrame.ARRAY-POSITlON,SIZE);
for (int i=O; i<SlZE; i++) {
        intArray.setValue(
                ScreenPanel.getRandom(10,100 ), i);
}
for (int i=SIZE; i > 1;  i- -) {
        for (int j=O; j< i-1;  j++) {
                if(intArray.getInt ( j ) > intArray.getlnt  ( j+ 1))
                        in tArray.swap ( j, j + 1);
         }
 }
}
```

Figure 1: JavaMy Language Code

The application in this research provides a built-in code editor or the user can write the code in any text editor of their choice. Once developed the user can build and run the code to view the animation. The animation frame consists of visualizations of the algorithm along with the control panel and a textual view of the code. The control panel provides the user with the ability to run the animation as a continuous or systematic (step-by-step) animation. A slider bar provides access to the execution speed control. The JavaMy code is very similar to the Java programming language except that a class definition is not a requirement for proper execution. When developing code in the JavaMy language, the user can select which of the data structures they want to animate and the language provides the required data types. Once written the code is parsed and compiled and the animation runs. One limitation of this research's application is that it only contains code for the most common data structures.

A unique technique used by Zhou, Li, Xian, Lai, and Liang (2008) attempts to provide a

solution to the reuse of algorithm system implementations using a context-aware methodology. Algorithm contexts hide its execution and the change in contexts drive the visualizations making an algorithm implementation independent of its animation and thereby creating a reusable system. Zhou et al. reference an important task to consider when developing an algorithm animation system as the specifications on how the visualization is connected and applied to the algorithm (Kerran & Stasko, 2002). Two approaches suggested by this research are event-driven and data-driven approaches. Event-driven techniques typically identify interesting events that correspond to relevant actions of the algorithm visualized by the approach. These interesting events turn into graphical events that execute as animation routines. Data-driven techniques use a mapping approach of the computational states into the graphical scenes. This approach uses a technique that declares attributes of object code to depend on variables of the program code. Stasko's (1995) POLKA libraries were also used in this research to provide examples of how current animation systems identify events and then compare this technique to their own context model. Zhou et al.'s results provide three advantages to using a context-aware method over an implicit approach:

1. Algorithm implementation is independent of algorithm animation.
2. Middleware for supporting the interaction between algorithm and animation is easier to develop and reusable.
3. The algorithm context model is reusable.

They conclude by stating that their context model is maintainable and reusable for a wider range of algorithm animations. Future work would include the development of an algorithm and animation repository and better design of the middleware to support more complex context management.

Human Computer Interaction (HCI) – Interface Design

Shneiderman, Plaisant, Cohen, and Jacobs (2009) suggest a sample of interface design guidelines to use when initializing development of a user interface:

1. Navigating the interface
2. Organizing the display
3. Getting the user's attention
4. Facilitating data entry

Navigation is essential to user comfort and there are several rules to keep in mind as the interface materializes: standardize task sequences, be descriptive when using embedded links, headings should be unique, radio buttons work well for single choice items, develop printable pages, and thumbnail images provide quick references for larger images.

Effective display organization may require the design to follow one or many of the following principles (Smith & Mosier, 1986):

1. Standardization of display data
2. Familiar features for display of information
3. Don't require the user to remember in between screens
4. Consistent input / output of information
5. User control of display data

User attention can be difficult to attract especially if there might be information overload or complicated design strategies. To minimize this interaction and maintain user attention, Shneiderman et al. recommend the use of these techniques:

1. Use high intensity sparingly with only two intensity levels
2. Use markings where appropriate

3. Larger sizes attract more attention, use up to four sizes
4. Up to three fonts
5. Add inverse coloring
6. Don't use blinking displays
7. Add up to four standard colors
8. When using audio choose soft tones and avoid harsh sounds

Data-entry is another important activity that should be considered in user interface design. When designing this component considered using similar action sequences, less user input actions, minimize user memory load, input / output of data should be linked, and allow user control of data display.

With respect to the design tools for user interface development Shneiderman et al. suggest using the "*Java Look and Feel Design Guidelines"* (Sun Microsystems, 2001) as a reference on user-interface design. These guidelines provide assistance for designers and developers to apply Java classes to develop consistent, compatible, and easy to use applications. They use the Java Foundation Classes (JFC) to provide a pluggable Java look and feel architecture that assists the developer in Graphical User Interface (GUI) design focusing on: design considerations, visual design, application graphics, and behavior.

Visualization may be defined in application development as "a graphical representation of data or concepts" or an "external artifact supporting decision making" (Ware, 2000, p.2). It is an important consideration when developing applications for human use as it assists with analysis and concept understanding by representing information visually. This technique of assistance is called cognitive support. Tory and Moller (2004) explore current human factors research as it applies to concept visualization. They introduce two new concepts called continuous model

visualization, that covers the visualization of algorithms that use a continuous model of the data and the discrete model visualization. Discrete model visualization refers to the algorithm visualizations that use discrete data models. Tory and Moller augment their research with a connection between visualizations and human factors-based design. Human factors-based design involves the use of design principles that focus on usability and usefulness of the intended user group. The researchers claim this concept is overlooked when developers initiate interface design and should be incorporated into applications that require users to view information or manipulation of data. A human factor can be physical or cognitive property of a single entity or social group but with regard towards visualization tends to be more cognitive. HCI methodologies that provide possible approaches for human factors-based design include:

1. User motivated design
2. User and task-based design
3. Perception and cognition-based design
4. Prototype Implementation
5. Testing – functionality and ease of interaction
6. User-centered design

Quantum Theory for Computer Scientists

Physicists endeavor to understand quantum mechanics completely but for computer scientists to contribute to the theory of quantum computation only a partial understanding of quantum mechanics is required. The question becomes "how much knowledge of quantum mechanics is enough?" Mermin (2006) describes two reasons why computer scientists can quickly understand the necessary quantum mechanical principles. He first describes a quantum computer as being a simple example of a physical system that is discrete, made up of a finite

number of units, and whose behavior is constrained and specified. Focusing on this two-state system can help avoid the peculiarities of quantum mechanics but should provide enough understanding relevant to basic quantum computation.

The most difficult part about learning quantum mechanics is the understanding of the difference between the essential and the inessential quantum phenomena that will be represented in an abstract model. Physicists take years to master this intuition but for an understanding of quantum computation, the only important part is the abstract model, or the product of the physicist's work. Mermin (2006) makes note of the fact that understanding quantum computation is simply the knowledge of the capabilities of the quantum computer not the need to understand how to build one.

A brief survey of several computer science programs concerning their introduction to quantum computer science curriculums find these popular textbooks as required reading for the course (Nielsen & Chuang, 2000; Mermin, 2006; Hardy & Steeb, 2001; Yanofsky & Mannucci, 2008). The researcher will refer the reader to these textbooks to provide a broader introduction on quantum mechanical concepts. This research will focus on the basic notions of quantum computing with particular emphasis on quantum algorithms, the fundamental concepts needed to introduce quantum algorithms to computer science students, and those concepts incorporated into the design of QuAL.

A Brief Survey of Quantum Mechanics

Quantum mechanical phenomena are difficult to understand since they exist at a molecular level and cannot be characterized by common experiences. Instead, mathematical formulas and natural associations create visualizations to assist with understanding these abstract principles and theorems. The best way to illustrate the fundamental concepts supporting quantum physics

is by using the two-slit experiment. Figure 2 displays a common depiction of the first setup using bullets.

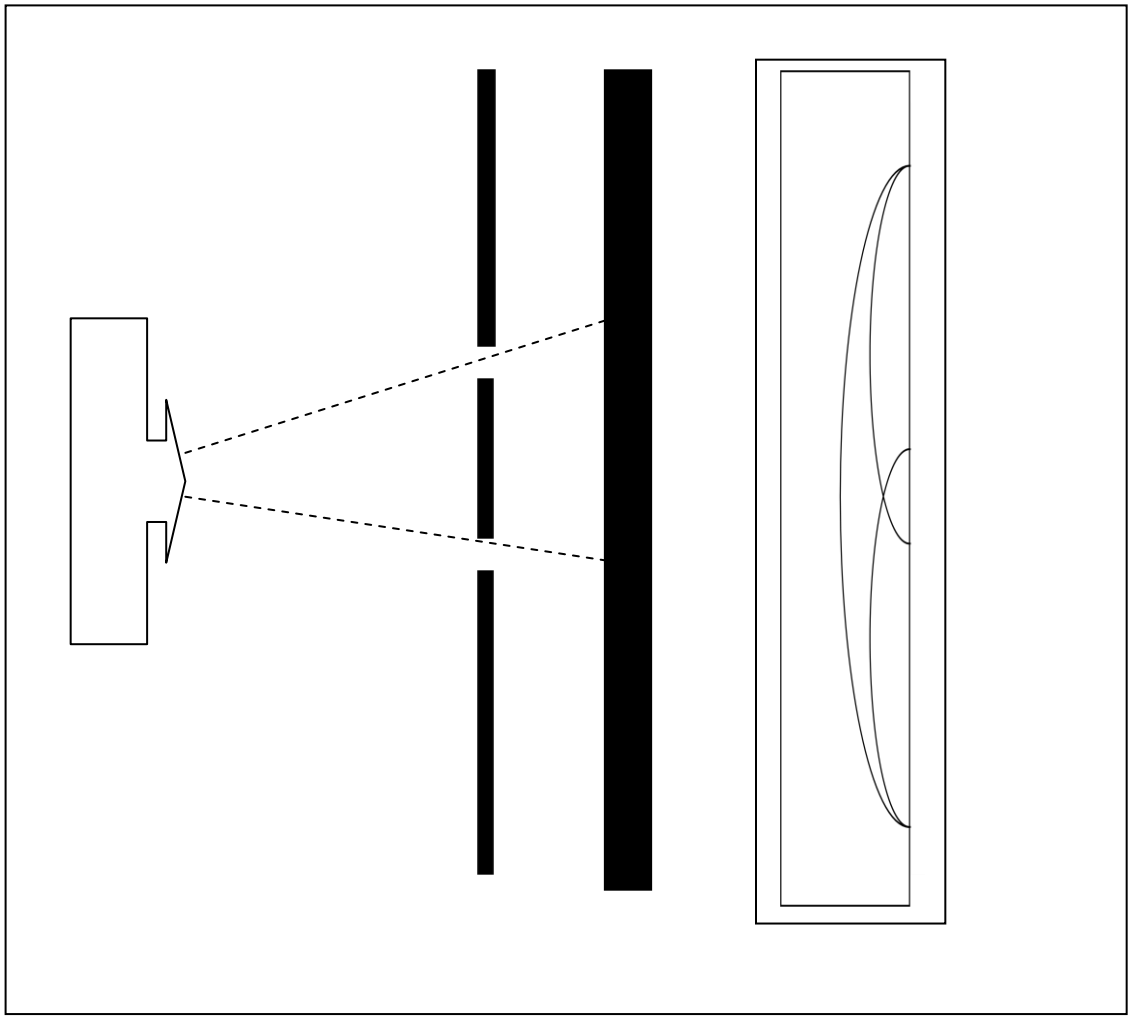


Figure: 2 Bullets – Two Slit Experiment

On the left, there is an object firing bullets at the wall with two slits. Behind the wall is another wall capable of measuring the number of bullets that go through the slits and measuring where they land. The graph to the far right will display the probability distribution for the bullets measured by the middle wall. In the first run of this experiment the top slit is opened, the graph shows the bullet's distribution is normal. Then the bottom slit is opened and the top slit closed. Once again, the graph displays a normal distribution. Finally, both slits are opened and bullets

hit the middle wall in a pattern that displays the distribution as the sum of the two previous distributions. The results show that the bullets go through one slit or the other in this demonstration of the two slit experiment.

In the next experiment, the object will emit sound waves instead of shooting bullets. Figure 3 illustrates this setup.

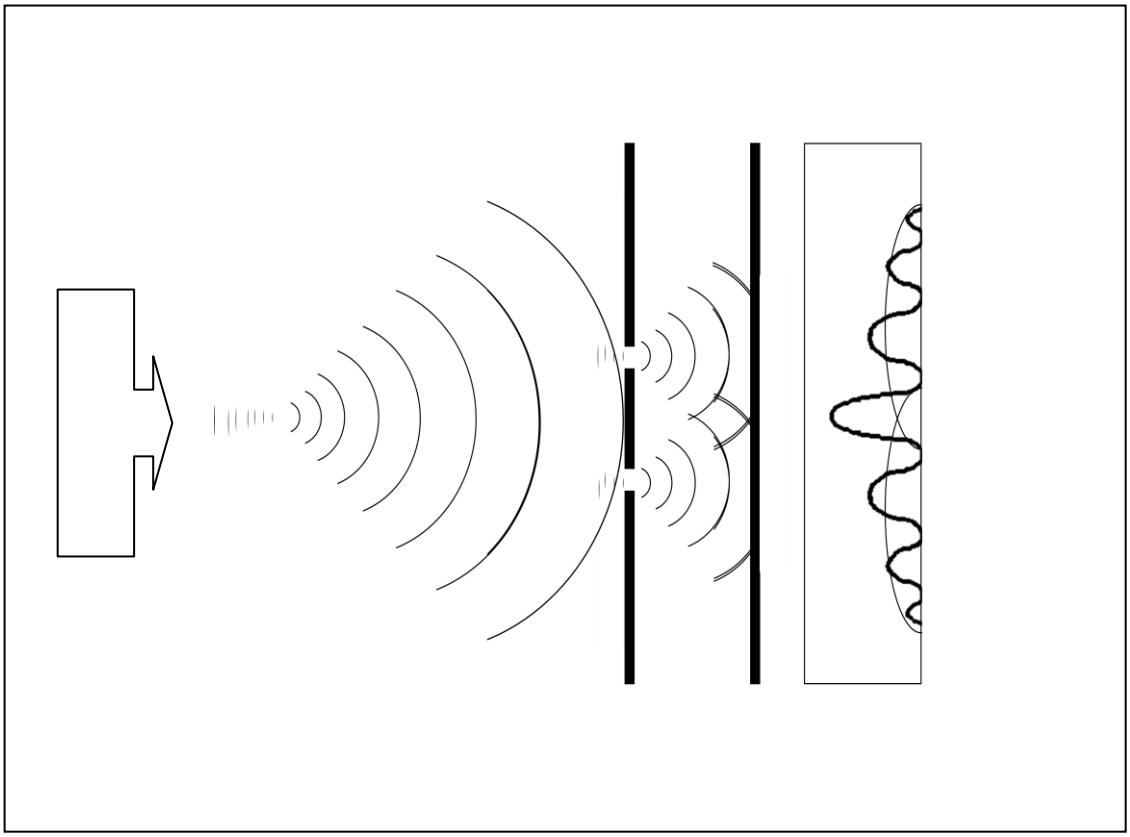


Figure: 3 Sound Waves – Two Slit Experiment

With sound waves the distribution is the same when only one slit is open but when two slits are open, the waves go through both slits at once interfering with one another and a sine wave distribution is measured by the middle wall. The next setup will examine the distributions of electrons as displayed in Figure 4.

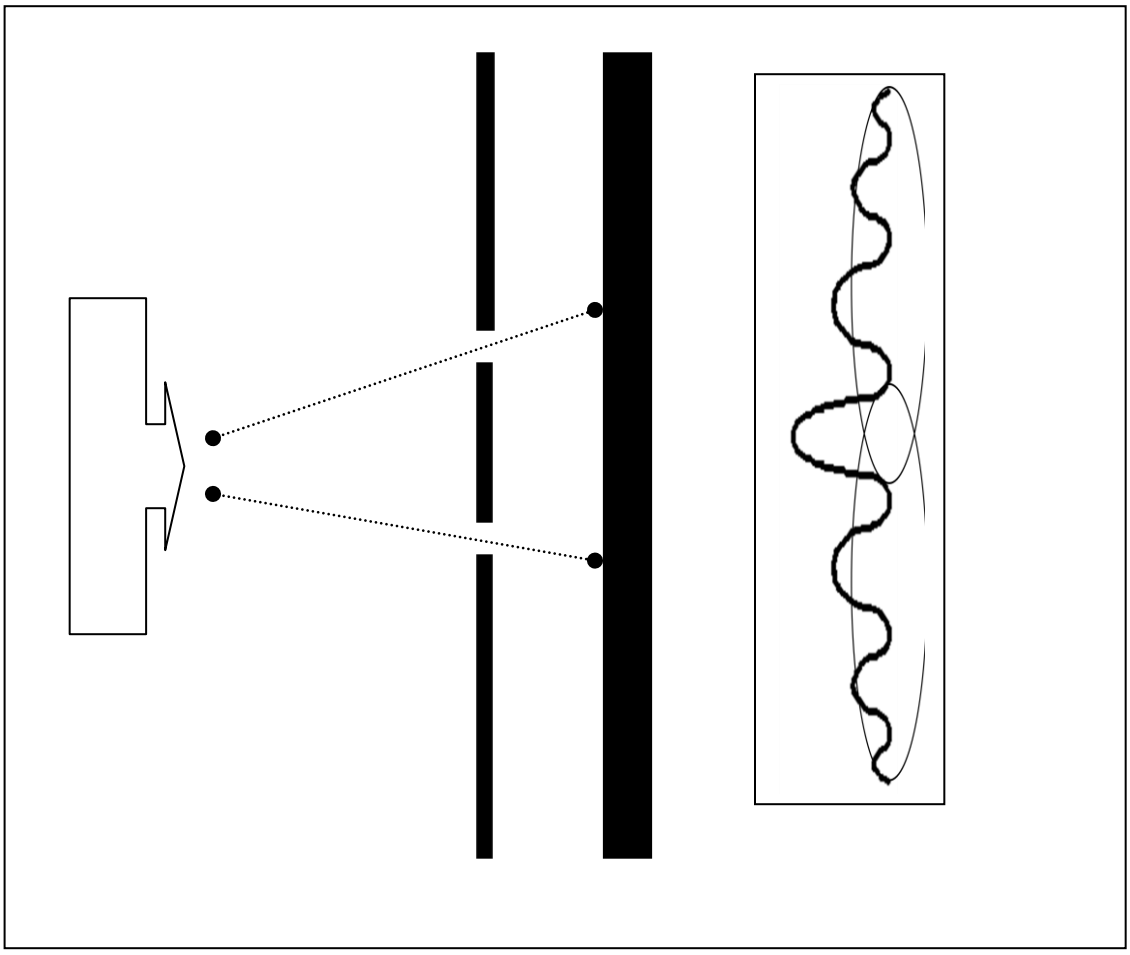


Figure: 4 Electrons – Two Slit Experiment

When only one slit is open the distribution is similar to the bullet's measurements shown in a normal distribution curve but when both slits are open, an unpredictable distribution occurs. The electron distribution is similar to the sine wave distribution. It appears that the each electron, shot one at a time to avoid interference, goes through the two slits at the same time and upon exiting, cancels itself out. This illustrates the quantum principle of superposition. The electron is not in a single position; rather it is in many positions at the same time. One limitation is that the electron cannot be viewed, as it exists in these multiple states. The middle wall measures the

electron's position and in doing so causes the positions or superpositions to interfere with each other, and collapse yielding a result. These elementary principles apply to the quantum bit or qubit, which is the basic building block of quantum information.

The mathematical formulas for this phenomenon are described in terms of the energy of a wave as measured by its height (Hey & Walters, 1987).

$$I = h^2$$

where *I* is energy and *h* represents the height of the wave. Energy depends on the square of the maximum height of the wave. Waves can fluctuate up and down, therefore *h* can be represented by a positive or negative value. When a wave goes through the two slits, the total disturbance h_{12}, when both slits are open, is calculates as the sum of the disturbances caused by the waves from each slit. Slit (1) is represented here as h_1 and slit (2) is h_2 so:

$$h_{12} = h_1 + h_2$$

Calculating the corresponding energy from the above formula gives us:

$$I_{12} = h_{12}^2$$

and then

$$I_{12} = (h_1 + h_2)^2$$

The wave amplitude is represented by I_{12}, expanding the right-handed component of this formula looks like this:

$$I_{12} = h_1^2 + 2h_1h_2 + h_2^2$$

Interference occurs when two slits are open as the waves go through each slit and recombine on the other side. This same mathematical principle can be used to describe the electron phenomenon that occurred in the two slit experiment except that the energy becomes the

probability of the electron arriving at a specific point on the middle detector wall and the height becomes the amplitude. *P* represents the probability and *a* in this formula represents the amplitude:

$$P_{12} = (a_1 + a_2)^2$$

The probability is related to the amplitudes as the sum of the amplitudes squared, which describes one of the basic principles of quantum computation.

The probability of an electron being at a certain point on the detector wall might be 20% at one point and 10% at another but quantum mechanics has shown in the previous illustrations that there is the potential of interference. In the quantum world, this means the potential of canceling itself out or a negative probability. The core of quantum theory explains this by using complex numbers that can cancel each other out and lower their probability. The probabilities of a quantum system are measured by the square of the amplitudes, which are complex numbers (Yanofsky & Mannucci, 2008). As an example, assume that there is a $1/\sqrt{2}$ chance of the electron being at point (0,2) and a $-1/\sqrt{2}$ chance of it begin at point (1,2). The square of $1/\sqrt{2}$ is $1/2$ and of $-1/\sqrt{2}$ is also $1/2$ so there is a .50 chance of it being at point (0,2) and a .50 chance of it being at (1,2). The probabilities add up to 1. Now using complex numbers the computation could be $i/2$ chances that it will be at (0,2) and a $-i/2$ chance of being at (1,2). The probability in this case would be the absolute value of the complex number squared or .25 in both cases. Quantum mechanics states that the probabilities of the states and transitions are complex numbers *c* such that $|c|^2$ is a real number between 0 and 1.

Quantum Bits

Quantum computing at its basic level is very similar to classical computing. The basic unit for classical systems is the bit, considered the basic building block of computing information.

The classical bit is considered to be in one of two states, either on or off, and is represented as a 0 or 1.

Quantum mechanical laws specify rules in quantum computation where a qubit is the basic building block of computing information. A qubit can be in a basis state of 0 or 1 but the difference and power of a qubit comes from the fact that it can also be in a superposition of states between 0 and 1. This means that a qubit can be in all possible states between 0 and 1 simultaneously. When the qubit is in superposition the measurement of this system will result in the collapse of the system to a single state. Before measuring the system, an outcome can predicted with some probability. The two computational basis states for a qubit are $|0\rangle$ and$|1\rangle$, expressed in Dirac notation or also called bra / ket notation. The ket notation $|x\rangle$ denotes column vectors or states and the bra notation, $\langle x|$ denotes the conjugate transpose of $|x\rangle$. A qubit is a unit vector in a two dimensional complex vector space with the state $|0\rangle$ represented by

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and the state $|1\rangle$ represented by

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and typically denoted by $\{|0\rangle,|1\rangle\}$. Qubits in superposition of the two basis states such as $a|0\rangle + b|1\rangle$ where *a* and *b* are complex numbers that specify the probability amplitudes such that$|a|^2 + |b|^2 = 1$, when measured will have a probability of being in state $|0\rangle$ as $|a|^2$ and a probability of being in state $|1\rangle$ as$|b|^2$.

A quantum state denoted in this paper as $|\Psi\rangle$ is a linear combination of all the states, $|x_0\rangle,|x_1\rangle,|x_2\rangle,...,|x_{n-1}\rangle$ with complex weights, $c_0, c_1, c_2, ..., c_{n-1}$ such that

$$|\Psi\rangle = c_0 |x_0\rangle + c_1 |x_1\rangle + c_2 |x_2\rangle + ...c_{n-1} |x_{n-1}\rangle.$$

or as:

$$\sum |c_i|^2 = 1$$

where $|c_i|^2$ is probability of finding each state. As an example, consider the following theoretical analogy, suppose our quantum system has three variables *f* = 0 if the *f* is false or *f* = 1 if *f is* true, and *b* = 0 if $b \neq 5$ or *b* = 1 if *b* = 5, and the third variable *s* = 0 if *s* is > 32 or *s* = 1 if *s* <= 32. Our state vector would be *[fbs]* with eight possible states since $2^3 = 8$ describing a system of n qubits having a state space of 2^n dimensions. If *f* = 1, *b* = 0, and *s* = 0 indicating that *f* is true, $b \neq 5$ and *s* is > 32, we would have the state [100]. We use only three numbers to define a state in this analogy. Next, suppose that we do not know what the exact values of *f*, *b* or *s* are at any given time. The probability of finding the describe state is:

.1[000] + .2[001] + .3[010] + .4[110] + .3[100] + .6[101] + .4[110] + .3[111]

This expression states that the complex weights are .1, .2 … to .3 for [111] and so the probability of getting the described state of [100] is $.3^2 = 0.09$. The sum of all the probabilities is exactly 1.0, which means that there is a 100% certainty that the described state exists. Unobserved, the values exist in all possible states but once the values are measures they collapse to one particular state. The assigned weights, provide a probability that our values will be in a specific state with [100] being measured 9% of the time and [001] 4% of the time. The exact state cannot be measured but a prediction of the probabilities can provide a possible solution. If one of the probabilities is near one and the others nearly zero, then the state nearing one will be considered a high-probability state and if one measures exactly one, then that state will be the one measured all of the time. A multiple qubit system such as the one describe above, can be assembled using

quantum registers similar to classical registers. Qubits are assembled into quantum registers with the tensor product expressed as:

$$\alpha = \alpha_0 \otimes \alpha_1 \otimes \alpha_2$$

for a three qubit system. So the state space for a three qubits, each with the basis $\{|0\rangle,|1\rangle\}$, has basis $\{|0\rangle \otimes |0\rangle,|0\rangle \otimes |1\rangle,|1\rangle \otimes |0\rangle,|1\rangle \otimes |1\rangle\}$ or expressed as $\{|00\rangle,|01\rangle,|10\rangle,|11\rangle\}$ more compactly. A basis state for a three qubit system is:

$$\{|000\rangle,|001\rangle,|010\rangle,|011\rangle,|100\rangle,|101\rangle,|110\rangle,|111\rangle\}$$

and in general an *n* qubit system has 2^n basis vectors. This exponential growth of the quantum system's state space defines the power of quantum computation.

A geometrical representation of a two-level quantum system is visualized with a Bloch Sphere:

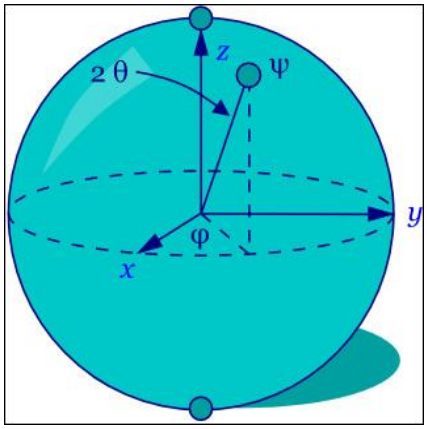


Figure 5: Bloch Sphere
(http://www.quantiki.org/wiki/index.php/Bloch_sphere)

According to Figure 5, only two real numbers are needed to identify a qubit, therefore we can map it to an arrow from the origin to the 3-D sphere of radius 1. Within this sphere context, qubits can be represented by two angles that correspond to the latitude (Θ) and the longitude (Φ) similar to a position on Earth. This quantum state would be represented by

$$|\psi\rangle = \cos\theta\,|0\rangle + e^{i\phi}\sin\theta\,|1\rangle$$

with

$$-\frac{\pi}{2} \leq \theta < \frac{\pi}{2}, \quad 0 \leq \phi < 2\pi.$$

In this notation, the North Pole represents the state $|0\rangle$ and the South Pole represents the state $|1\rangle$. The equator corresponds to a superposition of both the basis states with equal weights, and different phases. Representing a pure quantum state, the Bloch Sphere displays the fact that given any state on the sphere, the diametrically opposite points to one possible outcome of measurement on the system. This is a useful means of visualizing the state of a single qubit.

Quantum Entanglement

Entanglement is a quantum phenomenon that solves the measured outcome situation by claiming that the individual states of two particles are related or entangled, even if they are light years apart. Entangle pairs or EPR pairs are created and permitted to exist in a particular configuration, so if one particle is measured, the other one simultaneously yields a similar state. The benefit of this quantum principle is that a measurement can be extracted of the quantum system to form a basis without collapsing the system.

Einstein, Podolsky, and Rosen (1935) proposed an experiment commonly referred to as the EPR paradox. The implications of this experiment rocked the foundation of quantum theory at the time and reinforced the esoteric and unusual behaviors of quantum mechanics. This research paper contained the first definition of physical reality stating, in non-physicists' terms, that if a physical property of an object can be known without it being observed, then that property could not have been created by observation therefore if it was not created by observation, it must have existed as a physical reality before its observation. A common example used to visualize this

property is to consider two entangled particles $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, one is sent to observer Alice, and the other to observer Bob. Alice and Bob may be arbitrarily far apart. When Alice measures her particle and finds it to be in state $|0\rangle$ with a combined state of now $|00\rangle$, it is instantly known that Bob's particle will also be in state $|0\rangle$. If Alice were to have found her particle in state $|1\rangle$, then Bob would also have observed $|1\rangle$. An important note is that Bob has not actually measured his system but only observed it. Bob has not collapsed the quantum system state but when Alice measures her state both particles collapse. Quantum theory states that before Bob's system is measured, his particle can have no defined value for its state, it is in a superposition state. Only when measured does its value become physically real and this occurs when Alice measures her state. Einstein's explanation of this paradox was to assume Bob's particle possessed some kind of hidden fixed properties. The best that can be done to determine Bob's state is to give probabilistic predictions. This theory is commonly known as a hidden variable theory and claims that the simplest hidden variable theory for an EPR pair is that the particles are either both in state $|0\rangle$ or both in state $|1\rangle$. Einstein believed that there was no mysterious communication between the two particles; rather the particle properties were set when they were created. Bell's inequality experiment reveals the converse to be true. Bell proved that there is a strange connection between the particles, which instantaneously informs the undisturbed particle of the state change of the other entangled particle (Bell, 1964).

Quantum Decoherence

Quantum entanglement provided the proof that it is not possible to separate a particle being measured from the entity performing the measurement. In a quantum environment, the particle and its environment are bound together as one system. Both the measured particle and the

measuring device will have to be considered when attempting to understand and measure the quantum system.

The peculiarities of quantum mechanics create many questions affecting quantum computation. When a measurement of a quantum system is performed, what causes the quantum state to appear to "jump" to a particular eigenstate? Why can't a quantum system be viewed as a superposition of many other states? Recall the double-slit experiment described earlier in this section. In this experiment, the superposition of many other states is observed as constructive and destructive interference effects. Why do these interference states disappear in the quantum system?

First, recall how a quantum state is expressed as a linear combination of components of other states as the previous example showed here:

.1[000] + .2[001] + .3[010] + .4[110] + .3[100] + .6[101] + .4[110] + .3[111]

or expressed as

$$\psi_s = 0.5\psi_1 + 0.83\psi_2 + 0.24\psi_3$$

where a superposition state , ψ_s , is the sum of components of other states and their probabilities sum to one. In the double-slit experiment, interference components possess the same phase as they combine to produce the interference effects. This is also true for the superposition state, every component state must be in the same phase or coherent in order to achieve a superposition state.

In the real world, outside of our protected quantum environment, a particle is not completely isolated. It interacts with the environment that may have an effect of the particle 'being observed' by the environment and this could cause the quantum particle's component states to get entangled separately with different aspects of its environment. Potentially, each component

of the quantum particle forms separate entangled states causing their phases to be altered, which destroys the coherent phase. This principle is known as decoherence and describes how interference is lost. The interference components do not disappear but are simply out of phase and are not visual at the macroscopic level. Decoherence can be defined as the loss of information from a quantum system into the environment. As an example, imagine throwing a rock into the sea off the coast of Florida. After the initial splash, the waves dissipate and disappear from view. In actuality, they do not disappear but decrease in size, and get mixed into and interfere with other waves. Decoherence explains why states appear to jump since it happens so fast. It also explains why a superposition of states cannot be viewed, as decoherence has taken place long before measurement of the system.

Simple Quantum Gates

Up to this point, this section has described static quantum systems that change only when measured. Classical systems use boolean logical gates as a way of manipulating bits and in the quantum world, many of these same gates will apply with the exception that all operations that are not measurements are reversible and are presented by unitary matrices. These matrices represent quantum gates as counterparts to the classical gates similar to the logical AND or OR gates. Quantum gates can be visualized as rotations of the quantum state on the Bloch Sphere. Examples of common single-qubit quantum state transformations are displayed in Figure 6:

$$I:|0\rangle \rightarrow |0\rangle \qquad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$|1\rangle \rightarrow |0\rangle$$

$$X:|0\rangle \rightarrow |1\rangle \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$|1\rangle \rightarrow |0\rangle$$

$$Y:|0\rangle \rightarrow -|1\rangle \qquad \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$
$$|1\rangle \rightarrow |0\rangle$$

$$Z:|0\rangle \rightarrow |0\rangle \qquad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
$$|1\rangle \rightarrow -|1\rangle$$

Figure 6: Common Single-Qubit State Transformations

I is the identity transformation, which does not change its inputs. *X* is the negation transformation, *Z* is a phase shift operation, and *Y* is a combination of *Z* and *X*. *X*, *Y*, and *Z* are known as the Pauli Gates. Pauli-X gate is the quantum equivalent of a NOT gate and acts on a single qubit. Pauli-Y gate also acts on a single qubit and equates to rotation of pi around the Y-axis of the Bloch Sphere. Pauli-Z gate equates to the rotation of the Bloch Sphere around the Z-axis and acts on a single qubit.

Another important single qubit transformation is the Hadamard Transformation defined by (Rieffel & Polak, 2000):

$$H:|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

This transformation is used generate a superposition of all 2^n possible states which can be viewed as the binary representation of the numbers 0 to $2^n - 1$ (Rieffel & Polak, 2000).

$$(H \otimes \mathrm{H} \otimes ... \otimes H) \,|\, 00...0\rangle$$

$$= \frac{1}{\sqrt{2^n}} ((|\,0\rangle + |\,1\rangle) \otimes (|\,0\rangle + |\,1\rangle) \otimes ... \otimes (|\,0\rangle + |\,1\rangle))$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2n-1} |\, x\rangle$$

All five of the above transformations are unitary. Let M^* define a conjugate transpose of the matrix M, the definition of unitary for any matrix M states that if M is unitary, then $MM^* = \pm I$. Unitary transformations are considered as being rotations of a complex vector space. If M is filled with only real numbers, then M^* is just M^T, or the transpose of M.

Quantum Parallelism

In addition to entanglement, quantum computers are faster at some computations due to another quantum mechanical principle called quantum parallelism. True quantum parallelism is defined as the ability of a quantum computer to perform multiple computations simultaneously (Deutsch, 1985). It arises from the ability of a quantum memory register to exist in a superposition of base states. It is different from classical parallelism in that classical computing requires multiple processors linked together to perform parallel operations while the processors are performing other computations as well. The computational space increases exponentially with the number of particles in a quantum system, contrary to classical systems where an exponential decrease in time requires and exponential increase in the number of processors. This is an important concept to quantum algorithm development as it represents the fundamental

difference between classical and quantum computing (Kasivajhula, 2006).

Many quantum algorithms take advantage of quantum parallelism to achieve the desired results with high probability. The algorithms take advantage of the principle that allows quantum computers to evaluate a function *f(x)* for many different values of *x* simultaneously. Two common techniques used by quantum algorithms are (Rieffel & Polak, 2000):

1. Amplify the output values of interest or the 'marked' value.
2. Find common properties of all the values of *f(x).*

The next subsection provides examples of how quantum algorithms manipulate quantum parallelism principles to solve many computational problems.

Quantum Algorithms

Reversibility, superposition, and parallelism are the three major differences between classical and quantum operations (Omer, 1998). Reversibility is important in quantum computing as it provides the ability of the quantum operations to harness the power of quantum computation. Any classical subroutines performed in a quantum computation must be performed reversibly. Decoherence is a problem associated with quantum computers so to keep quantum computation coherent, quantum registers must be isolated avoiding entanglement with the environment. Deterioration of a quantum system has to remain constant since no heat dissipation is possible, therefore state changes have to be adiabatic. This means that with rapidly varying conditions, a quantum mechanical system must adapt its functional form. For this reason, all quantum computations must be reversible. Theoretical physics states that every operation on quantum bits must be undoable meaning enough information must be kept to work any operation backwards. Typical classical operations such as relational equivalence, Boolean AND, and Boolean OR, have to be modified for use in quantum computing. The quantum gates described

in the previous subsection provide a straightforward formula for converting irreversible classical operations to quantum operations. In addition, the NOT and CNOT gates provide classical Boolean functionality on a qubit.

Superposition and parallelism have been discussed in detail earlier in this paper and their importance to quantum computing explained. The ability for quantum registers to be placed in a superposition of states defines a powerful principle used by many quantum algorithms. Exploiting this quantum mechanical principle places a quantum bit in two different values at the same time. Quantum parallelism provides the ability to apply all the basis vectors in superposition simultaneously and supports the generation of a superposition of all the results. In this way, it is possible to compute *f(x)* for *n* values of *x* in a single application of a unitary transformation.

These quantum principles are exploited by the following quantum algorithms either all together or separately to solve intractable problems simply. When developing a quantum algorithm, a researcher has to think in terms of probabilistic factors, a conceptual change for many computer science programmers.

Deutsch's Quantum Algorithm

Deutsch (1985) developed one of the first examples of how a quantum algorithm can exploit quantum computational power. He presents an efficient quantum solution to a simple problem that requires an exhaustive search to solve deterministically without error on a conventional computer. In the Deutsch problem, a black box quantum computer is used to implement the function *f:* $\{0, 1\} \rightarrow \{0, 1\}$ in which the input is known to produce one of two outputs: balanced or constant. The function *f(x)* takes a one bit argument and returns one bit but can only be evaluated not altered as the definition of *f(x)* is immutable. One more criterion is that *f(x)* will

only be evaluated once. Classically, this problem would require at least two evaluations.

The Deutsch algorithm is based upon a quantum version of the classical Fourier transform that maps one complex-valued function of a real variable to another. This transformation typically maps the time domain to the frequency domain. Fourier transforms map functions of period *r* to functions that have non-zero values only at multiples of the frequency $\frac{2\pi}{r}$. On a classical computer, this operation takes time $O(n \log n)$ but on a quantum computer can take time $O(\log^2 n)$.

Four possible functions fit the requirements of the Deutsch problem:

1. *f(x)* → 0 //constant zero result
2. *f(x)* → 1 //constant one result
3. *f(x)* → x // identity function
4. *f(x)* → ~x // boolean negation

The first two are constant, which means they output the same value regardless of the input values. The last two are balanced because the output is zero half the time and one the other half of the time. Superposition and parallelism are the key quantum principles used by the Deutsch algorithm to determine whether a function is balanced or constant. Two qubits would be required putting four basis states: *f(00), f(01), f(10), f(11),* into a superposition to start, with the goal to determine whether *f(x)* is balanced or constant. The amplitudes start with positive values where *f(x)* = 0 and negative amplitudes where *f(x)* = 1. Recall that the sum of the squares of the absolute values of the amplitudes always sums to one. A quantum boolean exclusion operator is used to move each input value to and output qubit basically by swapping basis vectors around without changing the amplitudes. After this operation, there are still two positive values and two negative values. Quantum parallelism principles allow a query to all possible outputs

simultaneously or the combined state as a single operation. The resulting output would either print balanced or constant depending on which *f(x)* started as input when the quantum system is finally measured.

Simon's Quantum Algorithm

Simon's periodicity problem is a generalization of Deutsch's problem but instead of assessing whether a function is balanced or constant; it finds patterns in functions (Simon, 1997). Simon's algorithm is another black box problem that demonstrates quantum algorithms advantage over classical algorithms but does not solve a very difficult problem.

Assume a function $f : \{0,1\}^n \to \{0,1\}^n$ that is evaluated not altered in the black box quantum computer. The function f obeys the property that there exists a string $c = c_0 c_1 c_2 ... c_{n-1}$ such that for all strings $x, y \in \{0,1\}^n$, we have $f(x) = f(y)$ if and only if $x = y \oplus c$, where $\oplus$ is the bitwise exclusive-or operation. This means that values of *f* repeat themselves in some pattern and the pattern is determined by *c*, which is known as the period of *f* (Yannofsky & Mannucci, 2008). Simon's algorithm will solve the problem by finding the string c using a combination of classical and quantum procedures. An example of a function that satisfies the property follows assuming $n = 3$:

x	*f(x)*
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

Consider $c = 110$. Every output of f occurs twice, and the two input strings corresponding to any one given output have bitwise exclusive-or (XOR) equal to $c = 110$. Illustrating these

calculations:

$000 \oplus 110 = 110$; so, f (000) = f (110).
$001 \oplus 110 = 111$; so, f (001) = f (111).
$010 \oplus 110 = 100$; so, f (010) = f (100).
$011 \oplus 110 = 101$; so, f (011) = f (101).
$100 \oplus 110 = 010$; so, f (100) = f (010).
$101 \oplus 110 = 011$; so, f (101) = f (011).
$110 \oplus 110 = 000$; so, f (110) = f (000).
$111 \oplus 110 = 100$; so, f (111) = f (100).

Hadamard operations prepare the initial state and then call the black box to transform the state. Hadamard transforms convert the state performing simultaneous measurements on both registers. Given enough values, the algorithm can solve the *n-1* basis vectors, and compute string *c* that satisfies $c \cdot y = 0$, where *y* is the string measured by the quantum operations.

The classical operations are involved in the post-processing calculations where a classical algorithm uses the quantum results to solve linear equations. A pointwise exclusive or is used to add the binary strings and conclude the final result of *c*.

Shor's Quantum Algorithm

Simon's algorithm is of little practical use but does provide an exponential speedup over any classical algorithm. Its importance is often realized as the inspiration to Shor's polynomial-time factorization algorithm that stimulated the field of quantum computing.

Shor's (1994) factoring algorithm is a significant foundational quantum algorithm as given a practical quantum computer, Shor's algorithm would make many present cryptographic methods obsolete. The algorithm is probabilistic and based on the fact that the factoring problem can be reduced to the problem of finding the period of a function. Quantum parallelism is used in Shor's algorithm to obtain a superposition of all the values of the function similar to Simon's algorithm, in one step. A quantum Fourier transform, similar to the classical Fourier transform,

puts all the amplitude of a function into multiples of the reciprocal of the period. The resulting measurement of the state that yields the period is found with high probability and is then used to factor an integer or long list of digits. One issue with Shor's algorithm is that since the quantum Fourier transform is based on the fast Fourier transform, the process gives only approximate results in many applications.

Similar to Simon's algorithm, Shor's process uses both quantum and classical operations. Rieffel and Polak (2000) provide an outline of Shor's algorithm where *M* is the number factored:

1. When two numbers are coprime it means their greatest common divisor is 1. The first step in Shor's algorithm is to determine if *M* is a prime, an even number, or an integer power of a prime number. If yes to any of these then classical solutions are more efficient, there is no reason to use Shor's algorithm.
2. If no to any of those in step 1 then a quantum solution is more efficient and so step 2 states to obtain classically a random integer *a*, that is the power of 2 such that:

$$M^2 \leq a < 2M^2$$

3. Still using classical methods find an arbitrary integer *p* such that *p* and *M* are coprime.
4. Using quantum parallelism, compute $f(x) = a^x \bmod M$ for all integers from 0 to *(a-1).*
5. This step will prepare the amplitude function for use in step 7. In order to use the quantum Fourier transform, a state is constructed whose amplitude function has the same period as *f.* To accomplish this step, a random value *r* is obtained using the measurement of the qubit register from step 4. The value of *r* is inconsequential in itself. The importance is the effect of the measurement on the set of superpositions. What this means is that after measurement of this first qubit register we obtain a value for *r* and the other qubit register still in superposition when plugged into function *f,* will produce *r.*

Since $a^x \bmod M$ is a periodic function, it is known that the unmeasured superposition state after measurement is

$$C\sum_x g(x)\,|\,x, r\rangle$$

for some scale factor *C* where

$$g(x) = \left\{ \begin{array}{l} 1 \text{ if } f(x) = r \\ 0 \text{ otherwise} \end{array} \right\}$$

6. Now apply a quantum Fourier transform to the state obtained in step 5.
7. Measure the state of the qubit register left in superposition to extract the period of function *f*, calling it *m*. *m* has a very high probability of being a multiple of *a / t*, where *t* is the desired period.
8. Using *m*, and moving over to a classical environment, use the Euclidean algorithm to efficiently check for a non-trival common factor with *M*.
9. *Repeat the algorithm, if necessary.* Shor claims that a few repetitions of this algorithm yield a factor *M* with high probability.

Grover's Quantum Algorithm

Grover (1996) developed a less spectacular but easier to implement quantum algorithm that has many applications to database theory. He proposed an efficient solution using quantum concepts to the searching problem, although there are other applications of this technique. Grover's algorithm is also probabilistic similar to Shor's, meaning that the probability of failure for both algorithms can be decreased by repeating the algorithm. (Yanofsky & Mannucci, 2008) state that the number of repeats is significant and proven to be $\sqrt{2^n}$ times for Grover's algorithm.

Grover's algorithm returns to the black box context similar to Deutch's and Simon's previously defined algorithms. This searching problem is unstructured, which means that no

assumptions are used on the function *f*. A structured search, like searching an alphabetized list, is where information is known about the search space and *f*. Consider a function $f:\{0,1\}^n \to \{0,1\}$ that is implemented by a reversible transformation. Since this search is unstructured, there are no promises on the function *f*, so it is not possible to use a binary search or other methods to efficiently solve this classically.

Grover's algorithm is simple to implement with the following steps:

1. Begin with an *n*-qubit register in the starting state of $|0^n\rangle$.
2. Apply a Hadamard transform $H^{\otimes n}$ on the *n*-qubit register.
3. Repeat $\sqrt{2^n}$ times:

    3a.) Apply phase inversion operation: $U_f(I \oplus H)$

    3b.) Apply the inversion about the mean operations: $-I + 2A$
4. Measure the qubits.

Summarizing the implementation of Grover's algorithm, step one and two are standard operations for many quantum algorithm's, which first initializes a qubit register to a starting state and then using the Hadamard transform puts the qubit register into a superposition of all input states. Step 3 contains the difficult task of attempting to obtain a useful result from the superposition. The trick is to change the quantum state so as to greatly increase the amplitude of the marked state and decrease the amplitude of all other states to improve the probability of finding the correct search element. The phase inversion operation should provide the initial state change and inversion about the mean should amplify the marked state moving its probability higher with each iteration of the loop in step 3. If all works correctly step 4's measurement should provide the correct solution in the least amount of time. Classically repeating a search

process over and over should provide a better solution but Grover's algorithm can provide worse results if executed too many or too few times.

Wallace and Narayanan's (2001) used a derivative of Grover's algorithm in their quantum sorting algorithm that is animated by this research. Steps 3a and 3b are discussed in detail in Chapter 4 of this final report, as Grover's algorithm is important to the design and implementation of QuAL.

A Survey of Other Interesting Quantum Algorithms

Current quantum research has taken these foundational algorithms and modified them to achieve greater efficiency, enhanced them to solve broader classes of problems, or adapted them to other types of problems. Brassard and Hoyer (1997) propose a new quantum algorithm that combines the techniques of Simon and Grover's algorithms to solve a decision problem in exact quantum polynomial time. Quantum computation theory challenges the corollary of the Church-Turing thesis that states that anything that can be computed in polynomial time on a physical device could be computed in polynomial time on a probabilistic Turing machine. Simon's algorithm provided a solution that may be solved in polynomial time on a quantum computer, but its classical counterpart would require exponential time when the data was supplied in a black box. Exact quantum polynomial time pertains to problems that quantum computers can solve in guaranteed worst-case polynomial time.

Buhrman et al. (2001) provide a generalization to Brassard, Hoyer and Tapp (1997) algorithm, which was one of the earliest applications of Grover's (1996) algorithm. Buhrman et al. utilize quantum amplitude amplification to solve the element distinctness problem. The problem of elemental distinctness attempts to find out if all the elements in a list are distinct. The classical version of this algorithm can provide a solution in $\theta(n \log n)$ while Buhrman et al.'s

solution can solve the problem faster in $\theta(n^{3/4})$ queries. Ambianis (2005) developed an even faster solution for this problem in $\theta(n^{2/3})$ queries by using quantum walks. Quantum random walks have been a very popular concept in developing new quantum algorithms. Physicists first described them in 1993 (Aharonov, 1998) and they became an important computational tool when interest in quantum computers increased. Quantum random walks provide illustrations for the quantum concept of interference, which is one of the concepts that computer scientists should understand when attempting to understand quantum algorithm development. The quantum walk problem is similar to classical random walks, which uses the analogy of a person walking along a straight line and assessing the direction of their next step. In the classical version, a probability distribution describes the walker's current state that refers either to the fact that they have a choice to go forward or backwards with equal probability. In the quantum walk version, the walker is in a superposition of all positions measured by the amplitudes instead of the probability of taking a step forward or backwards. Recall that amplitudes in quantum computing measure the probability of being in one state or the other and in this analogy, they determine the walker's next step but it is not just forward or backwards but a superposition of both choices.

Mathematically, because amplitudes use complex numbers, this could mean that our directions cancel each other out. Typically, in a classical situation, the probability of reaching zero position is the sum of the two probabilities with equal probability of going in two different directions, but in the quantum world, research has found a different scenario. Amplitude measurements do not have to be positive numbers. A negative result may occur causing a number greater than unity. This effect is known as quantum interference. Ambainis, Kempe and Rivosh (2005) provide a solution to the problem of interference by using a quantum coin flip in their quantum walk algorithm. The quantum walk's performance is improved to $\theta(\sqrt{n})$ searching

a spatially d-dimensional space. Quantum walks have also been used to improve upon triangle finding and verifying matrix products (Magniez & Santha & Szegedy, 2005; Bhurman & Spalek, 2005).

Shi (2001) uses concepts from Grover's (1996) algorithm and Ambainis' (2000) research to prove that $\Omega(n \log n)$ comparisons are necessary for quantum sorting algorithms that use only comparisons. Ambainis' previous lower bound was $\Omega(\log n)$. Shi's improved lower bounds mean that the best comparison-based quantum sorting algorithm can be no better than a constant time faster than its classical counterpart can. In another paper, Shi (2002) worked to improve searching, sorting and element distinctness with quantum concepts. Brassard et al. developed a quantum algorithm that traverses a binary search tree using quantum routines more efficiently than classical algorithms. Once again using the black box model so that the only way the algorithm can obtain information about the input data is via queries. The use of binary search trees is different from others who base their quantum algorithms on Fourier transforms and amplitude amplification (Brassard, Hoyer, Mosca & Tapp, 2000).

Using the sequential quantum circuit's model, Klauck (2003) examined the complexity of sorting proving that quantum sorting algorithms are more efficient in a space bound setting. Klauck confirmed this theory by claiming that the quantum complexity of sorting is different from classical complexity. Using Grover's search algorithm and Durr and Hoyer's (1999) quantum algorithm for finding minimum, Klauck demonstrates the following theorem: *"For all S in* $[\Omega(\log n),...,\theta(n/\log n)]$ *there is a quantum circuit with space S that, given a comparison oracle for n numbers, outputs the sorted sequence, and uses time* $\theta(^{3/2}\log^{3/2} n/\sqrt{S})$. *The entire output is correct with probability 1 -* $\in$ *for an arbitrarily small constant* $\in \succ$ *0."*

This research focused on Wallace and Narayanan's (2001) algorithm to animate the quantum sorting concept. The main purpose of using a sorting quantum algorithm was the connection to classical sorting concepts and the use of comparisons to bridge the gap between lecture and animator usage.

Wallace and Narayanan (2001) use superpositional permutation searching to propose two new quantum algorithms for sorting and routing. The sorting quantum algorithm uses input of an unsorted list of n items in random order to obtain an output of a sorted list of n items in a specific sequence by using a derivative of Grover's search algorithm. Their research uses an alternative approach to the classical sequential inspection and rearrangement, by recasting the sort process as a search for a particular permutation of the list items sequenced in the desired sorting order amongst all possible permutations of the list items. This quantum algorithm is discussed further in Chapter 4 along with the derivative of Grover's algorithm used in their research.

Summary of Knows and Unknowns

Quantum algorithm research is expanding its boundaries from the early historical discoveries with the innovation of new algorithms, enhancements to foundational quantum algorithms and broader problem solutions. Whether a quantum computer will emerge as the next new technological revolution has yet to be determined but quantum computing concepts are furnishing improved solutions to classical problems and computer scientists are beginning to realize the value of these concepts. Quantum theory lags behind universal theory but quantum computer scientists are hoping for a time when they can develop quantum algorithms that will someday run on a universal quantum computer.

Historical classical animator research has covered a broad range of theoretical topics and proven that with certain parameters, animations are an effective means of conveying algorithm

concepts. Techniques such as speed control, visualizing code sequence, addition of colors, along with supplemental lecture material were used by this research to contribute to QuAL's aesthetic enrichment. QuAL's interface followed a few foundational guidelines by providing familiar features, user control, consistent input / output, and added inverse coloring.

Mermin (2006) claims that computer scientists do not need to understand how to build a quantum computer but should comprehend the quantum mechanical properties required to interact with one. Developing new quantum algorithms may require knowledge of entanglement, quantum superposition, qubits, amplitudes, and distribution probabilities. An introductory understanding of these properties could entice student interest in quantum computing and promote its usage into more computer science applications.

Chapter 3

Methodology

Research Goal and Design Objectives

The primary goal of this research was to design and develop an application that animates quantum algorithms. Computer science students to promote learning quantum concepts and entice student interest in quantum computing can then use this animator in an introductory computer course. A design objective for this research was to develop an application that has an easy to use interface providing animations of quantum concepts such as; entanglement, probability distribution, amplitudes, qubits, quantum states, and superposition. Another design objective was to provide animator features that promote a clear understanding of the steps required by a quantum algorithm to use quantum concepts in solving a sorting problem. Sorting concepts were used to promote a connection between a student's classical algorithmic knowledge and the quantum concepts promoted by these animations.

Three stages were used in this research; an exploratory stage, a design and development stage, and finally the case study. The exploratory stage gathered data from classical algorithm animation research and compiled a list of common features that could be used in the design of QuAL. Quantum algorithms have similar operations to classical algorithms by requiring an input, stepping through code statements, and then producing an output. Many of today's quantum algorithms will use principles designed by classical algorithms but enhance those procedures with quantum mechanical principles to provide faster, more efficient solutions. Wallace and Narayanan's (2001) algorithm animated by this research utilized classical sorting

concepts to check the final product produced by their sorting quantum algorithm, testing to assure ascending sort order. The main difference between classical and a quantum algorithmic procedure is that you cannot see or measure the operations of a quantum algorithm. They operate in a blackbox environment and only when observed does the state collapse to a specific measurement. As an example, the common bubble sort algorithm can be animated by visualizing blocks of different heights moving from one position to the next as the algorithm processes the ascending sort order. The quantum sorting algorithm cannot be measured during the sort operation so the procedures themselves cannot be animated during the sorting process. The quantum concepts can be animated by checking quantum states or amplitudes and probability distributions as the algorithm's methods calculate them.

Exploring Classical Algorithm Animators

The literature review provided research topics and data covering many classical algorithm animators. The past and present research focused on user interfaces, design features, animation theory, and algorithm understanding. Concepts and components that worked for the classical environment may not provide enhanced interaction or design improvements in the quantum environment due to the differences in procedures between the classical and quantum algorithms.

The exploratory study discovered animator strengths and shortcomings. Animations developed based on just aesthetics instead of concentrating on what a student needs to aid understanding failed in student comprehension of presented material. Algorithm animation is also plagued by the simple fact that animating abstract data has many different possibilities depending on the requirements of the algorithms. Multiple views can also complicate learning by becoming too confusing for the student but need to be used in order to display all the required concepts for algorithm understanding. The user should not be left wondering why an animation

is moving, what values is it attempting to display, and how did the algorithm accomplish its final steps. This research found that effective animators used many of the following findings: textual annotations to assist students in associated mapping, expression of conspicuous or important features, explain cause and effect clearly, dynamic instead of static images using clear visualizations of cause and effect, multiple representations, and factual representations to reduce complexity (Grillmeyer, 2001).

This research found that no one path or environment used in the classical algorithm animation realm would completely define the initial design of QuAL and its features. The abstruse principles of quantum mechanics could not be animated by simple classical algorithm animations but the animations could be defined by the complex numbers (amplitudes) and probability distributions calculated during the course of the quantum algorithm's execution. QuAL's initial design would incorporate text and graphical chart representations showing algorithm progress described by Wallace and Narayanan's (2001) pseudocode, highlighting each statement as it progresses through its operations, using a bar chart to represent amplitude changes and a line chart plotting the dynamic activity of the probability distributions. Textual changes are incorporated into this initial design to display important concepts associated with specific quantum algorithm sorting processes.

QuAL's interface was designed to provide a clearer, more accessible representation of the information and interactions occurring in the animations. Different colors portray the dynamic changes occurring with the probabilities and the color red was used to highlight the 'marked' amplitude. A speed control slider provides the user with a chance for slowing or speeding up of the animations and moving the control to the left end will slow the speed enough to follow each animation systematically. Multiple views can be confusing but the user can gain an appreciation

for the algorithm's processes by examining one view at a time and then a combination as animator understanding increases. The input window is non-editable in this initial prototype to control qubit register growth, future versions will allow user input. The interface background colors white, light grey and blue are used to provide a calm learning environment as quantum concepts can be frustrating to learn.

QuAL: Design and Documentation

The programming design of QuAL used an object-oriented development approach (OODA), which followed the traditional five phases of analysis, specification, design, implementation, and evolution. These phases provided feedback into one another in an iterative approach as changes in design and specifications were required during implementation. Figure 7 provides a diagram of this iterative process:

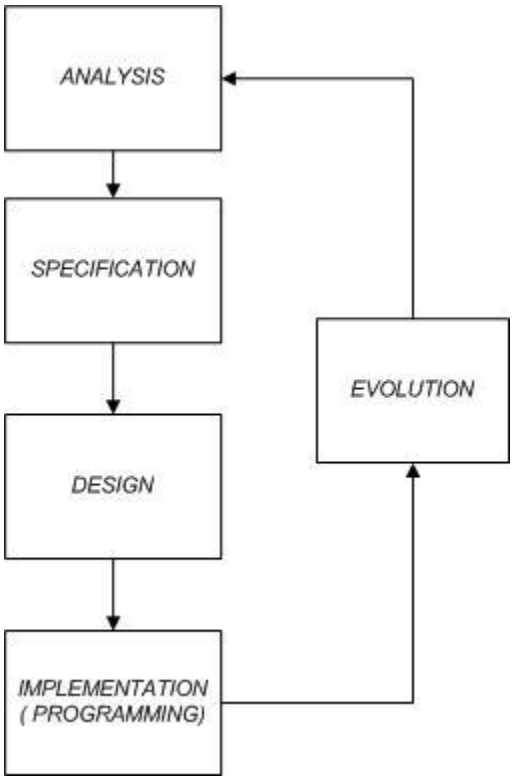


Figure 7: OODA Process

During the OODA process, the Object-Oriented application framework was used to guide the development of QuAL's graphical user interface (GUI) using the Java programming language. (Laszlo, 2002) furnishes a survey of framework characteristics stating one of the main benefits of using object-oriented code is its support for reuse. Other benefits of using a framework for the development of GUI-based programs are:

1. Frameworks provide generic components allowing the developer to focus on other application features.
2. The developer will customize the framework by adding application-defined classes that connect to the framework.
3. Frameworks are responsible for flow of execution focusing the development effort on other design elements.
4. The framework assists the development process by defining interfaces for components providing the developer with a mechanism for customizing new components but still may use the components provided by the framework.
5. The framework may contain design patterns.

QuAL's Class Diagrams and Summaries

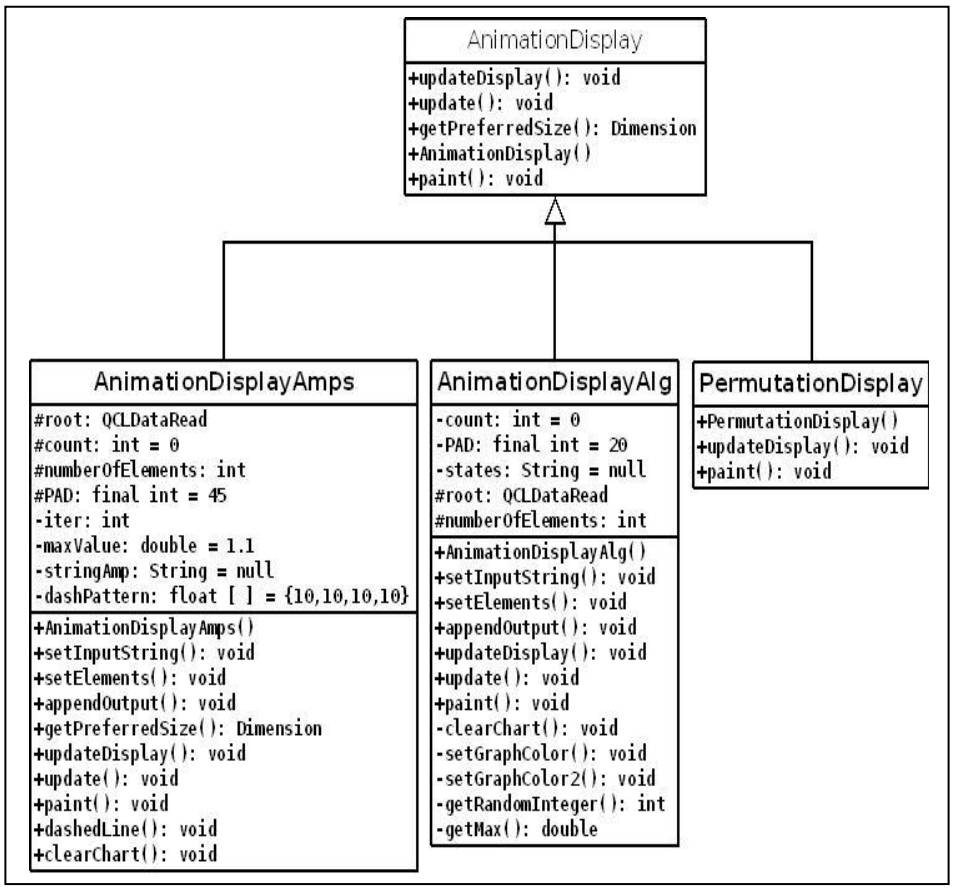


Figure 8: UML Class Diagram for QuAL's Amplitude and Probability Animations

Figure 8 shows several of QuAL's interesting animation classes. This class diagram generated with Dia, represents the class AnimationDisplay and its subclasses AnimationDisplayAmps, AnimationDisplayAlg, and PermutationDisplay. AnimationDisplay is an abstract class that extends Canvas and is defined within the core package. It holds the basic display methods for QuAL's animations. The three subclasses extend its parent class and further define methods specific to the types of animations they represent. AnimationDisplayAmps contain the methods and attributes for animating the quantum sorting algorithm's amplitude data, AnimationDisplayAlg holds the methods and attributes for animating the probability data and

PermutationDisplay will hold future methods for adding more animation specific to the different permutations. Appendix K provides a complete UML class diagram containing all of QuAL's classes and their relationships.

The next class diagram displayed in Figure 9 shows the parent and child classes responsible for the control of the animation and main graphical user interface.

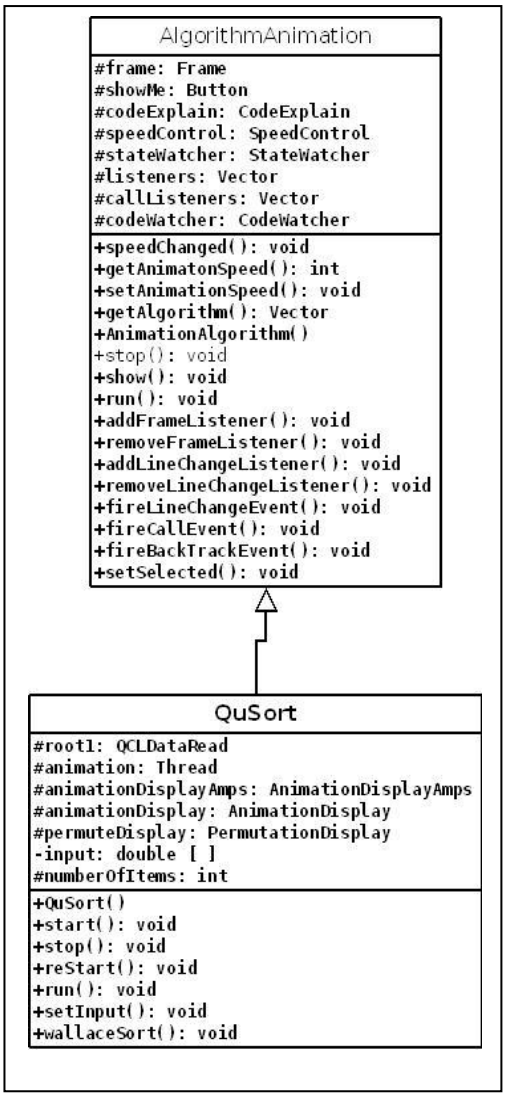


Figure 9: UML Class Diagram of Animation Control and GUI Classes

QuSort extends AnimationAlgorithm to control the coordination of the different animation views

and contains the initial setup for the main graphical user interface. It initializes methods in the parent class AnimationAlgorithm upon execution of the Java Applet with the location of the textual file containing the quantum sorting algorithm's pseudocode and other essential starting values. It also defines the wallaceSort method used to provide feedback and fire events when the algorithm updates probabilities and amplitudes and sends that information to the class called AnimationAlgorithm that coordinate the algorithms state with each of the views.

The AnimationAlgorithm class implements runnable and SpeedChangeListener as the main animation control class. Its methods control animation speed, determine which algorithm is animated, and initializes the speed control, line change, and frame listeners. It contains methods that watch for events as the algorithm's pseudocode highlighting is changed and coordinates the changes of those lines of code. The initial prototype of QuAL animates the Wallace and Narayanan (2001) sorting quantum algorithm but this system has been developed so that future versions could quickly introduce other algorithms via the addition of new packages. The main package core, shown in Figure 10, contains the foundational classes for the animation of quantum algorithms. A second package core.quantSort, displayed in Figure 11, contain the classes specific to animating the Wallace and Narayanan sorting algorithm. Additional classes could be added to animate other algorithms by simply adding a new package and adding a switching component to the main interface allowing the user to select between different algorithms. Future additions to QuAL could animation other sorting algorithms or other classes of quantum algorithms.

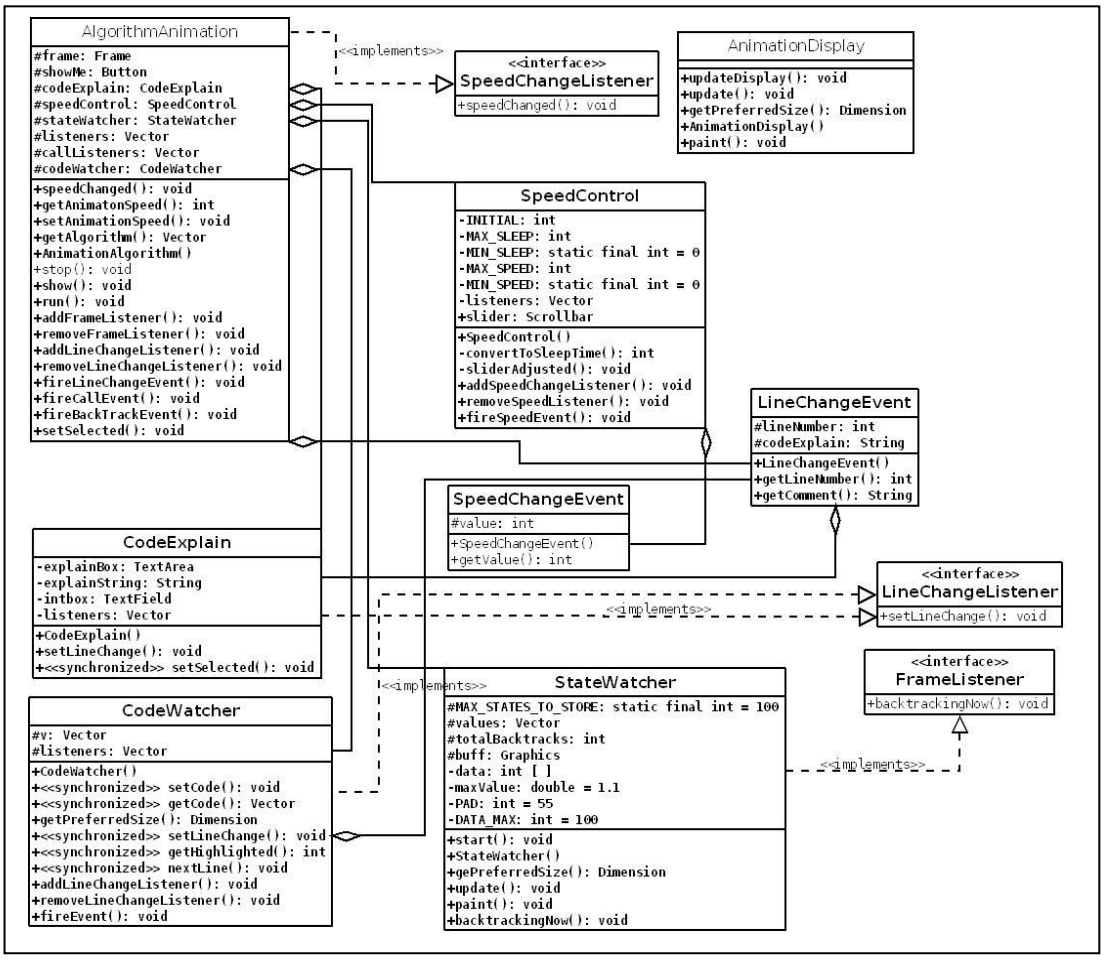


Figure 10: core Package Classes

The constructor of the AnimationAlgorithm class initializes listeners as vectors and sets up initial values for the CodeWatcher, CodeExplain and SpeedControl classes. The AnimationDisplay class is an abstract class that extends Canvas and its bare methods provide a basic starting point for the graphical setup of the animations. LineChangeEvent is used to control the movement from line to line of the highlighted pseudocode text and to coordinate the addition of comments in the text box provided during animation execution. The CodeWatcher

class implements a LineChangeListener and its methods provide updates to other classes by exchanging messages between the classes. The SpeedChangeEvent class extends EventObject and controls the value to increase or decrease the speed of the animation if the speed control slider is modified during algorithm execution.

QuAL's listener interfaces are implemented by several classes and provide the simple task of maintaining the line-changing event, the speed-changing event and the future addition of backtracking to add the reversible functionality to QuAL. The ability to go backwards is not functional in this initial prototype but may be added to future additions to allow the user to work backwards through algorithm execution.

The SpeedControl class contains the layout instructions for the speed control slider and its methods are responsible for calculating the time to sleep if the speed is increased or decreased, adding a speed change listener or removing it, and firing a speed event when the slider is moved. The CodeExplain class implements a LineChangeListener and contains the methods that setup the components for the textual changes in the code text box. This class also sets the static values for non-editable input values of "6, 2, 9, 1" which are the integers sorted by the quantum algorithm. In future versions this class will contain the text boxes that allow the user to enter different and additional integers to display. The LineChangeListener coordinates the changing of the code text box with the highlighted pseudocode changes.

In Figure 11, AnimationDisplayAlg class is responsible for the probability distribution animation view. This class extends the abstract model class AnimationDisplay with the implementations of paint( ) and updatedisplay( ), coordinating changes of the probabilities with the amplitude view and code highlighting view. It contains methods that control the randomization of colors for the probability changes assisting the user with viewing the subtle

changes as the algorithm executes its code. The paint ( ) method uses a PAD variable set to control the height and width of the animation so that is maintains an appropriate size as the animation changes and values increase and decrease. AnimationDisplayAmps is similar to the AnimationDisplayAlg class except it contains the code required to display the amplitudes and their changes during algorithm execution. The PermutationDisplay class is a bare bones class intended to display permutations changes in future versions of QuAL.

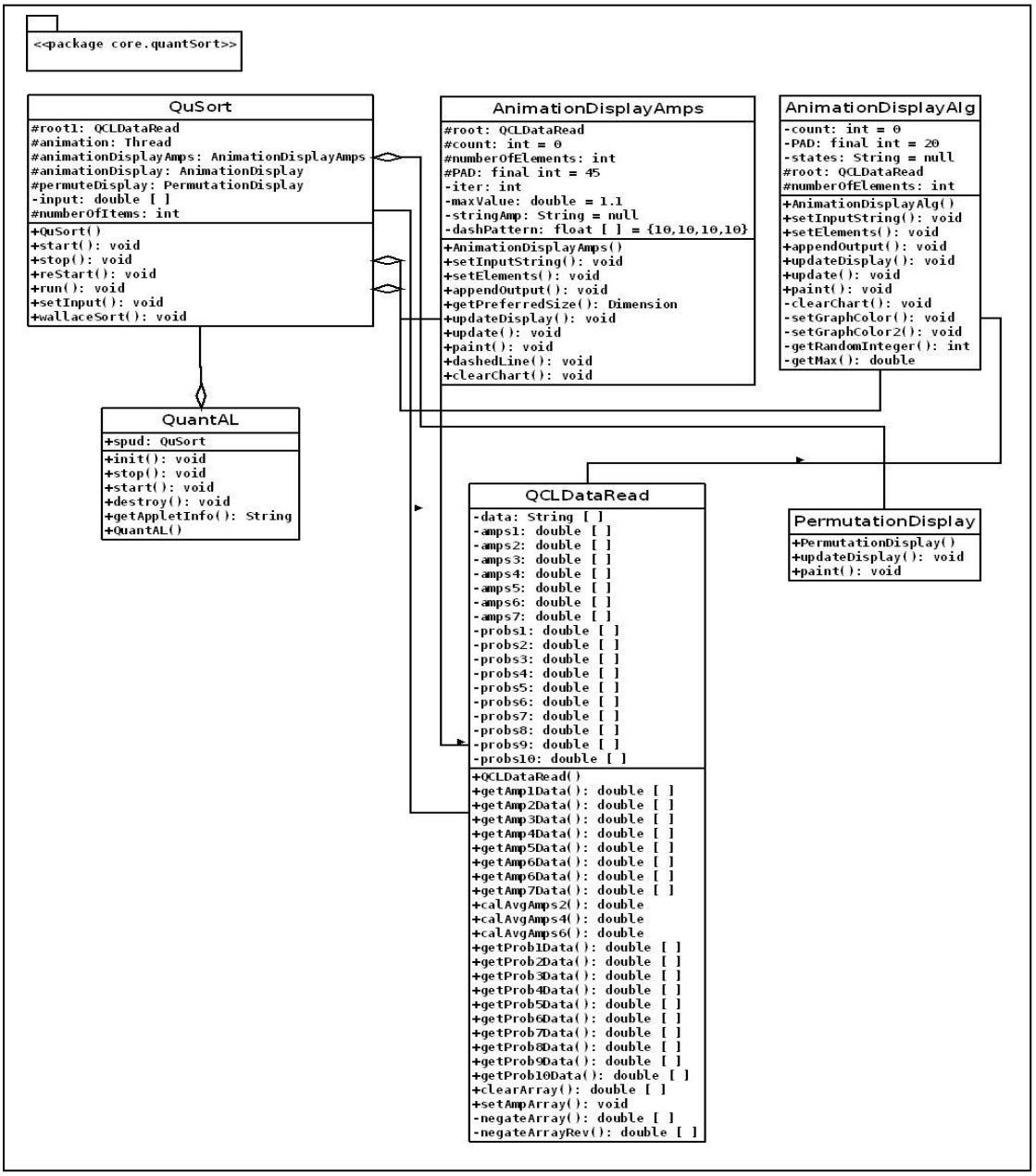


Figure 11: core.quantSort Package Classes

QuSort and QuantAL classes are the main applet classes. QuantAL contains the applet initialization methods that start the applet with a small popup window that automatically starts the applet but if the user stops the applet it displays a button to restart the animation if needed. QuSort describe above extends AnimationAlgorithm and initializes classes as it coordinates the flow of data collected from QCLDataRead.

The data gathered during the exploratory stage provided options for quantum algorithm development as well as classical animator information. Initially a simulator was not going to be used in the design of QuAL but development of quantum data structures were cumbersome and the Java programming language did not support some of the functionality required to develop quantum algorithms. Bernhard Omer (1998) developed the Quantum Computation Language (QCL) as part of his Ph.D. work at the Technical University of Vienna. QCL was selected due to its easy installation in a Linux environment and since it functioned as an interpreter, starting the application could be achieved at any location on the server by simply typing the command ‘qcl’ at the command prompt.

The data class QCLDataRead was added in order to utilize QCL. QCLDataRead’s main purpose is to parse the data generated dynamically as the algorithm executed its procedures. Initially code was parsed via the Java file I/O procedures using streams and channels until it was executed as a Java applet and web browser security issues forced a change into using the class URL representing a Uniform Resource Locator. The URL object pointed to the file located within the same web readable directory on the server. Using the URL class and the StringBuffer class allowed the transfer of material to occur without security restrictions or causing browser

certificate failures.

Animation mapping in QuAL uses the data-driven approach that relies on the assumption that observing how variables of an application change provides information to the actions performed by the underlying algorithm (Demetrescu, Finocchi, & Stasko, 2001). This method worked well with the hidden components of quantum algorithms allowing QuAL to animate based on capturing and monitoring data changes rather than on processing interesting events.

QCLDataRead class parses the dynamically generated text files using a regular expression focused on finding the amplitude and probability calculations generated by calls sent to the QCL interpreter by a helper class called CmdExec. The CmdExec class opens an input / output stream sending commands to QCL and recording the data retrieved as the algorithm executes its procedures. The Wallace and Narayanan (2001) sorting algorithm was coded using QCL and stored as a .qcl file so that the CmdExec class could easily send commands to the QCL interpreter as strings. Typical classical algorithm animations record each step of the algorithm results in a transition from one state to another. The state is then mapped into a visual representation and usually shows the transitions as animations between these visualizations. QuAL and its quantum algorithm animation is designed using a similar process but instead of mapping the transition from one state to another it maps the data gathered by inserting a print command into the wallace.qcl algorithm file capturing, via stream output commands, the calculations. Data is stored in arrays by the QCLDataRead class and mapped to the animations views. Probabilities are displayed using the plot charts and amplitudes are displayed using a bar chart as the algorithm incorporates Grover's (1996) code to find the 'marked' value. A Perl script, called by CmdExec calculates the specific permutations of the values "6,2,1,9" and stores them in an array. The index of the "1,2,6,9" permutation indicates a sorted element and if it

matches the value sent as the ‘marked’ value to wallace.qcl then the animator knows the quantum algorithm found the correct sorted solution.

The Java Applet was compiled and runs on a Linux server running Debian 2.6.26-24. Java version 1.6.0.20, QCL version 0.6.3 , and Perl version 5.10.0 were used to develop and execute QuAL.

The initial proposal research intended to include the Template Method pattern into the design of QuAL as a method of defining an algorithm (Gamma et al., 1995). The Template Method would have executed the algorithm, which calls one or more hook methods. The pattern would not have achieved any benefit within the quantum algorithm environment since QCL provided the basic algorithmic design steps and the implementation was accomplished by the QCL interpreter.

The Case Study

The case study was performed in stage five of this research. Walsham (1993) claims that “the most appropriate method for conducting empirical research in the interpretive tradition is the in-depth case study” (p. 14). In the previous section, Creswell (2003) recommends a qualitative approach, including the use of case studies as a qualitative methodology. Yin (2003) writes that a case study is defined as “an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident” (p. 13).

Runeson and Host (2008) provide guidelines for case study research with reference to software engineering. They support the use of case studies claiming that it studies contemporary phenomenon in a natural context and expand the characteristics of research methodologies by using Robson’s (2002) classifications of the four types of purposes for research:

1. Exploratory – answering the question of what is happening, creating new insights and then generating ideas and hypotheses
2. Descriptive – portraying a situation or phenomenon
3. Explanatory – looking for an explanation of a problem
4. Improving – attempting to improve something of the studied phenomenon

Yin (2003) provides evidence that case studies are common to the fields of psychology, sociology, political science, social work, business, and community planning. These case studies are conducted to increase knowledge about many different entities and social behaviors or cultural events so it seems reasonable to compare these processes to the field of software engineering (Runeson & Host, 2008).

This research followed the suggested guidelines of Kitchenham et al. (1995) to plan and design, execute, and analyze the results of this case study:

1. Define the hypothesis
2. Select the pilot projects
3. Identify the method of comparison
4. Minimize the effect of confounding factors
5. Monitor the case study against the plan
6. Analyze and report the results

The main research goal was to build a quantum algorithm animator with a hypothesis that using it as a pedagogical tool will help computer science students understand quantum computing concepts more than lecturing to them. This case study used computer science students entering their sophomore or junior year of school to test this hypothesis.

Planning: The Subjects

This case study was carried out at Wayne State College in Wayne, Nebraska during the summer academic semester with the participation of 20 students from a variety of Computer Science courses. Participation in this research was strictly voluntary and none of the students were currently enrolled in classes instructed by this researcher. The participants were mainly male computer science majors within their first or second year as a major. This gender ratio is typical of many computer science courses taught here at Wayne State College. Participation was on a voluntary basis, and motivated by the prospect of curricular improvement.

Planning: The Object

QuAL is the developed quantum algorithm animator and will be the object used to determine if students learn more about quantum concepts than students who just listen to a quantum computing lecture. The case study aims at answering the following research questions:

1. Will using QuAL provide better post-test results?
2. Will the students consider QuAL helpful or user-friendly?
3. What quantum concepts will both groups grasp and what concepts will be improved by using QuAL?

Planning: The Project

Twenty students volunteered their time to participate in this case study. Pre-test containing questions about initial quantum computing and algorithm knowledge, began the study with every student taking the same test at the same time. A quantum computing lecture followed that presented a brief overview of basic quantum mechanical principles, quantum computing concepts, classical versus quantum comparisons, and information about foundational quantum algorithms. Students did not know which group they would be assigned to until the lecture was

completed in order to partially minimize the effect of confounding factors. Once they were assigned a group letter A or B, they were asked to move to a common side of the lab. Group B students were asked to leave the room for a short break while Group A students completed the post-test and exit survey. Once Group A finished completing the documents they were asked to leave the lab and Group B members were asked to return. Group B members used the QuAL tutorial and were allowed to interact with the animator and ask questions before taking the post-test. No time limit was specified and most students used another hour to interact with QuAL and ask questions about its operations. Group B was then asked to complete the post-test and exit survey. A hypothesis for this research was that algorithm animations should assist student learning so Group B students should achieve better scores on the post-test.

Execution: Methods of Comparisons

The comparisons were accomplished by using the pre-tests to determine initial quantum computing knowledge and comparing them with their specific post-test results. Data was also gathered comparing Group A pre-test / post-test results to Group B's pre-test / post-test results. Exit survey results were used to assess aesthetic, interface, and other helpful information about their experiences with QuAL.

Execution: Minimize the Effects of Confounding Factors

Knowledge of group assignments was addressed only after the pre-test and lecture was completed. This should minimize knowledge or assumption of group assignments that might introduce errors. Using summer session students who volunteered provided a normal distribution of student representation to minimize the effect of differences in response variables. A confounding factor that may introduce issues is the fact that Group B was not able to learn about the tool and evaluate the tool in separate activities.

Execution: Monitor the Case Study against the Plan

The study's progress and results were compared to the plan. The case study was executed correctly with no external factors causing result bias. All student volunteers remained as planned and executed the required steps as described, no changes were recorded.

Analysis: Data Collection

The results of this case study are presented in the next Chapter of this report. Collection of the data was accomplished by gathering the pre-tests / post-tests and exit survey once both groups completed them. Copies of the exit survey and the corrected pre / post-test are found in appendix F and G respectively. The tests contained all the same questions and were divided into the following four sections:

1. Basic quantum computing knowledge: questions 1,5,9,13,17
2. Quantum states: questions 2,6,10,14,18
3. Quantum sorting: questions 3,7,11,15,19
4. Quantum algorithms: questions 4,8,12,16,20

These sections were used to determine knowledge of a specific quantum concept in order to determine and cross-reference concept learning. Test questions were crafted based on findings from the exploratory study using questions common to classical algorithm animations and from research material presented to introduce quantum concepts. No partial credit was given as all questions have only one right answer. All the questions in the tests were multiple choice and were constructed to measure understanding of the concepts presented in the lecture. Both groups answered the exit survey but Group A only answered the first four questions. Group B answered all the questions, as 5 – 14 were specific to QuAL.

Summary

The design of the quantum algorithm animator (QuAL) was guided by data collected during the exploratory stage of this study. Several of the concerns addressed during the design and development stage, were to provide a clear mapping between accumulated data and animations, clearly illustrating the most noticeable aspects of the system, and reduction of complexity.

QuAL was implemented using a variety of methods including textual annotations, color, movement, multiple views, and code highlighting. All of QuAL's animations were designed to expand students' knowledge of quantum computing and entice interest in quantum algorithms.

The case study was performed at Wayne State College in Wayne, Nebraska and results were compiled and assessed. All of the data was collected as written test results then tabulated and graphically presented using PASW Statistics 18 release 18.0.0 and Microsoft Excel.

Chapter 4

Results

Introduction

This chapter covers an in-depth look at QuAL's animations focusing on how they work, and how they illustrate the different concepts of quantum computing. The design and presentation of these animations were influenced by the exploratory study of past and present classical algorithm animators and on prior quantum computing work. The details of the design of these animations and QuAL's inner Java code interactions are given in Chapter 3.

QuAL was used in the case study introduced in Chapter 3 and the details of that study are analyzed and presented in this section. Students were divided into two groups with both groups listening to the quantum computer lecture and only one group using the animator. The instructional framework in which the students preformed the case study work was organized to simulate a common classroom setting at Wayne State College. Typical classroom diversity would be higher male to female ratio and would include lecture, lab, and time for questions. The case study volunteers were all 18 years or older and majored in computer science entering either their sophomore or junior year in the program.

The quantum sorting algorithm animated was Wallace and Narayanan's (2001) algorithm that used principles of Grover's (1996) foundational searching quantum algorithm to locate the permutation matrix that represented an ascending sort order of four integers. The sorting problem should be a familiar classical concept to first or second year computer science students who should have a preliminary knowledge of the classical approach.

The Quantum Algorithm Animator (QuAL)

Quantum Sorting Algorithm

Wallace and Narayanan (2001) proposed an interesting method for sorting integers based on superpositional permutation searching, which uses Grover's (1996) foundational quantum algorithm's principles. The algorithm takes as input an unsorted list of n items in arbitrary order and outputs a sorted list of n items in a sorted sequence. The method used by Wallace and Narayanan's algorithm is a derivative of Grover's unstructured search algorithm with the assumption that a quantum gate *Q*, which implements the function *f(x)* is defined as:

$$Q_F|x, 0\rangle \rightarrow |\ x, f\ (x)\rangle$$

In a classical sort algorithm, a common procedure would be to inspect each integer and rearrange them based on a specific sort solution into a final sorted order. This algorithm uses an alternative approach using quantum mechanical principles to recast the sort process as a search for a particular permutation sequenced in the desired sort order. As an example, the following table is a listing of all the permutations of the integers 3,6,1,8:

3 6 1 8	3 6 8 1	3 1 6 8	3 1 8 6	3 8 6 1	3 8 1 6
6 3 1 8	6 3 8 1	6 1 3 8	6 1 8 3	6 8 3 1	6 8 1 3
"1 3 6 8"	1 3 8 6	1 6 3 8	1 6 8 3	1 8 3 6	1 8 6 3
8 3 6 1	8 3 1 6	8 6 3 1	8 6 1 3	8 1 3 6	8 1 6 3

Table 1: Permutations of 3,6,1,8

The number of permutations on a set of *n* distinct integers is given by n factorial *(n!)* so in this example with four integers there are 4! = 24 permutations of {3,6,1,8}. The marked permutation

illustrated above in quotes is the sorted permutation as its ordered set is {1, 3, 6, 8}.

Grover (1996) used quantum principles to produce a drastic improvement in searching an unstructured database for a specific item or multiple items. The key principle to Grover's faster search is the quantum principle of superposition. This quantum speedup occurs because a quantum computer can exist in more than one state at a time and in Grover's case search different parts of a database at the same time. A quantum computer would need only 1,000 steps to find a correct solution in a database with a million entries since

$$n = \sqrt{N}$$

where n is the number of steps needed by Grover's algorithm and N is the number of items in an unstructured database.

Grover's algorithm starts out by initializing a quantum register into a superposition of all possible items in a database. Observing the register at this point would reduce the probability of selecting the right answer although the register contains the right answer. Instead, Grover's algorithm involves a sequence of quantum operations on the register's state similar to the wave phenomena in quantum physics, that is that all the desired results will interfere constructively and all the others will interfere destructively cancelling each other out. By manipulating the phases using quantum computing operations, it's possible to obtain an estimate of the mean, derived from the quantum state of the entire system, in fewer steps than other algorithms. Although Grover's algorithm requires a quantum computer, these operations can be simulated on a conventional computer as this research has done by using the Quantum Computational Language (QCL).

Wallace and Narayanan's (2001) algorithm integrates Grover's procedures into their sorting algorithm when searching for the marked sorted permutation. The database described by

Grover's algorithm becomes a list of all n factorial permutations where one permutation represents a sorted list of integers. The listing of their quantum sorting algorithm can be found in the appendixes as well as in screenshots of the code highlighting view of QuAL. The quantum sorting algorithm starts off by initializing the number of qubits and iterations required to find a solution. Qubits were defined in Chapter 2 of this final report as the basic building blocks of quantum information. The maximum number of permutations determines the number of qubits required to sort four integers. Since 4! = 24 and 24 is not a power of two, the setup for QuAL will have to include 2^5 or 32 values in order to hold all of the permutations required to animate the sorting of four integers. The number of iterations is defined as $\sqrt{2^n}$ times. It has been proven that this is the number needed to provide the best possible solution as running the algorithm more times will reduce the probability of finding the correct solution as well as running it less times (Yanosfsky & Mannucci, 2008). Once these values are initialized two quantum registers are created to store the superpositional permutations and to hold temporary values during the quantum operations. The Wallace and Narayanan (2001) algorithm begins using Grover's (1996) principles by creating a superposition containing all the possible permutations of the original four integers {6,2,1,9} with their initial states (amplitudes) equally set to $1/\sqrt{32}$ or 0.17678. Superposition is created by using a QCL external subroutine creating a Hadamard gate, which can be generalized to:

$$H :| i\rangle \rightarrow 2^{-n/2} \sum_{j \in B^n} (-1)^{(i,j)} \mid j\rangle$$

The Hadamard gate is a generalized qubit rotation and defined by the transformation matrix:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

The QCL syntax looks like this (Oemer, 1998):

extern operator H(qureg q); // Hadamard gate

and sets the predefined qubit register *q* into a superposition of states. The main loop begins by running the query function. Figure 12 contains QCL's predefined query function for Grover's operations:

```
qufunct query(qureg x,quvoid f,int n) {
   int i;
   for i=0 to #x-1 {
          // x -> NOT (x XOR n)
        if not bit(n,i) { Not(x[i]); }
   }
   CNot(f,x); // flip f if x=1111..
   for i=0 to #x-1 {
        // x <- NOT (x XOR n)
        if not bit(n,i) { !Not(x[i]); }
   }
}
```

Figure: 12 QCL' s query

This query function allows formulation of the problem within the realms of classical boolean logic to solve the equation *f(x)* = 1. If the desired results are found, the quantum NOT gate interferes constructively, if they are not the desired results interference occurs destructively cancelling each other out and increasing the probability of a correct solution. The *Not( )* function flips the value of a bit while the *CNot (f, x)* function tests the value of *x* and if it's "1", it flips the value of *f*. The main loop of Grover's algorithm utilizes two other predefined procedures from QCL, a controlled-phase-gate:

$$CPhase :| x, y\rangle \rightarrow i^{x,y} \mid x, y\rangle$$

and a diffusion operator:

$$D = \sum_{ij} |\, i\rangle d_{ij} \langle\, j\,|$$

The *CPhase ( )* function takes a classical floating point number as its first operator and a qubit as its second argument. For this sorting quantum algorithm, the code will use a predefined constant pi as the floating point number and one of the qubit registers as the second argument. *CPhase (pi, f)* will alter the amplitudes of the machine basis states where *f* is $|1\rangle$ multiplying them by $e^{i\pi} = \cos\pi + i\sin\pi$ = *-1*, which flips the sign of the $|1\rangle$ component.

The diffusion operator uses the Hadamard Transform and the conditional phase rotation to perform another main quantum operation used by Grover. This operation is referred to as inversion about the mean or inversion about the average (Yanofsky & Mannucci, 2008). This operation boosts the separation of phases in which amplitudes from the unmarked element are transferred to the marked element. Yanofsky and Mannucci (2008) provide a good example explaining this operation in terms of a sequence of integers. Suppose we have five integers: 51, 37, 15, 20, and 72. The average of these integers is 39. We can represent these numbers as an image like this:

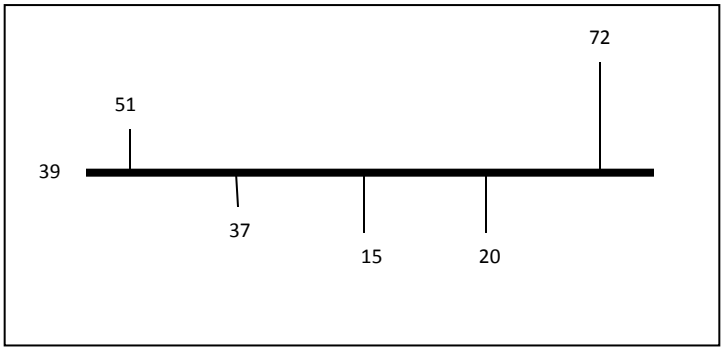


Figure 13: Integers and their Average

The image displays the principle that the average is the number such that the sum of the lengths of the lines above the average is the same as the sum of the lengths of the lines below. Grover's average about the mean then changes the sequence to a new condition where each element of the original sequence above the average would be the same distance from the average only below the line. This is also true for the elements below the line changing to their appropriate distance from the average but above the line. Mathematically we are simply inverting each value around the mean calculating them like this: 51 is considered to be 39 – 51 = -12 units away from the average so then by adding 39 to -12 we get the new element 27. The next element 37 would then become 39 + (39 – 37) = 41. This is the new image displaying all the changed elements:

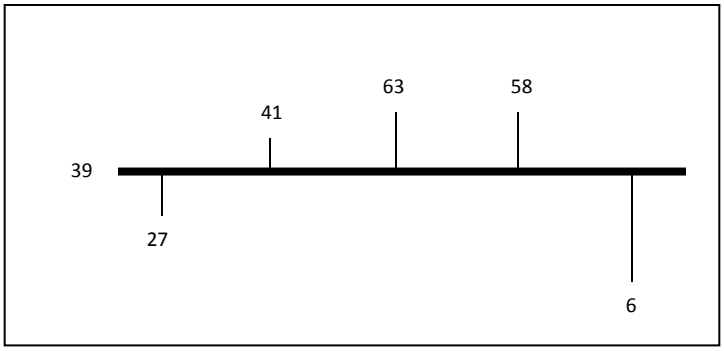


Figure 14: Average about the mean

Grover has formalized this process as $v' = -v + 2a$ with the average of the sequence remaining at 39 but the sequence changes to 27, 41, 63, 58 and 6. By combining the phase inversion and the inversion about the mean, the whole process becomes an effective operation separating the amplitude of the marked state from the unmarked states.

The procedure wallace ( ) was created from the grover ( ) procedure predefined by Omer (1998) as:

```
procedure wallace(int n) {
            int l = 5;   // no. of qubits
            int m=ceil(pi/8*sqrt(2^l));     // no. of iterations
            int x;
           int i;
           qureg q[l];   //setup qubit registers q and f
           qureg f[1];
  print "qubits:",l,":",m,":iterations";
  {
           reset;
           H(q);              // prepare superposition
           print "Initial Amps:";
           dump;
           print "Initial Probs:";
           dump q;
            for i= 1 to m {     // main loop
                      query(q,f,n);     // calculate C(q)
                      print "Pass#:",i,":probs:";
                      dump q;
                      CPhase(pi,f);     // negate |n>
                      !query(q,f,n);    // undo C(q)
                      print "AfterNegate:Amps:";
                      dump;
                      print "AfterNegate:Probs:";
                      dump q;
                      diffuse(q);       // diffusion operator
                      print "Diffusion:Amps:";
                      dump;
                      print "Diffusion:Probs:";
                      dump q;
           }
           measure q,x;       // measurement
           print "measured:",x;
  } until x==n;
  reset;              // clean up local registers
}
```

Figure 15: wallace () procedure

QuAL's wallace ( ) is the main function stored in the wallace.qcl file executed by QuAL to output data captured by CmdExec.java. As the data is calculated, QuAL utilizes the appropriate

amplitudes and probabilities to animate them in action displaying the increase in probability of the marked item and showing the phases of Grover's inversion about the mean operations.

QuAL's Animations

QuAL contains multiple views of the quantum sorting algorithm's operations. The first view is the textual display of Wallace and Naraynan's algorithm with the executing line highlighted to provide the user with an illustration of moving from statement to statement as the animations change. Here is a screenshot of the textual view:

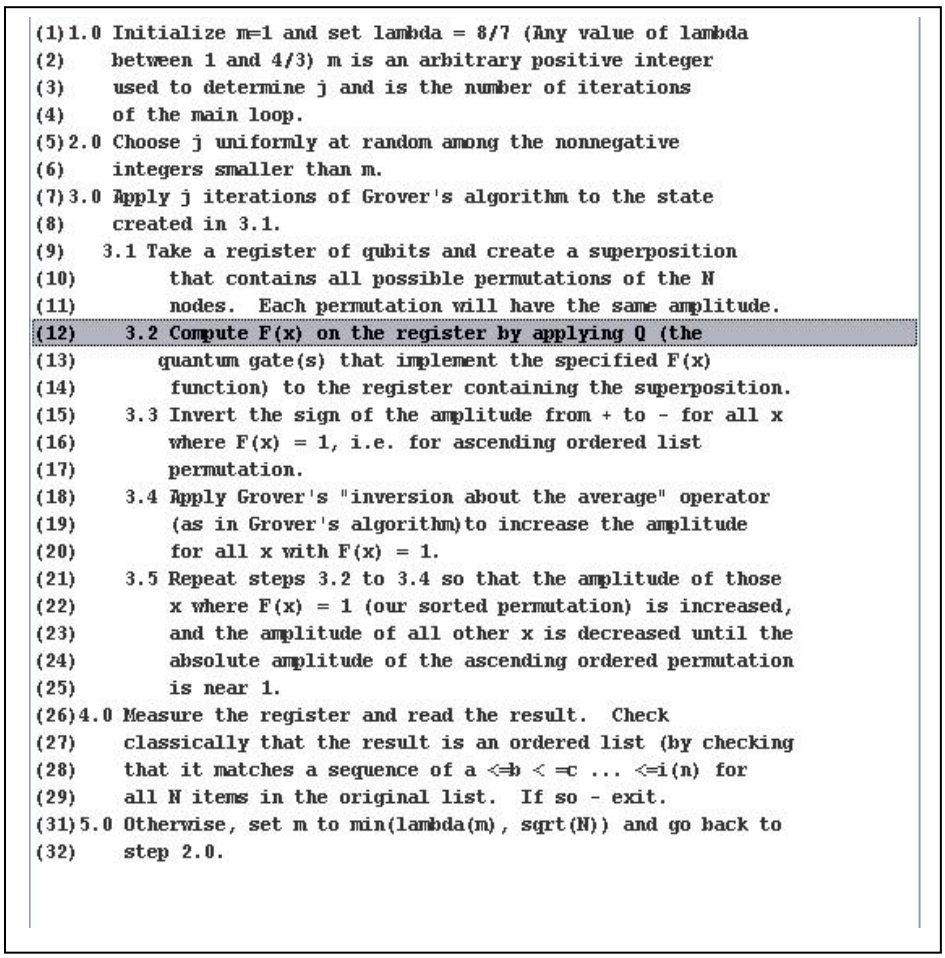


Figure 16: Code Highlighting View

The highlighted code is line number 12 in this screenshot. The highlighting moves from statement line to statement line as the algorithm executes the code and loops until the initialized value ends the procedures and the value is measured. The animator is setup to continue looping until the user exits the program. This allows the user a chance to consider all views as they execute or change the speed for faster or slower viewing.

The next view is the animation of the probability distributions:

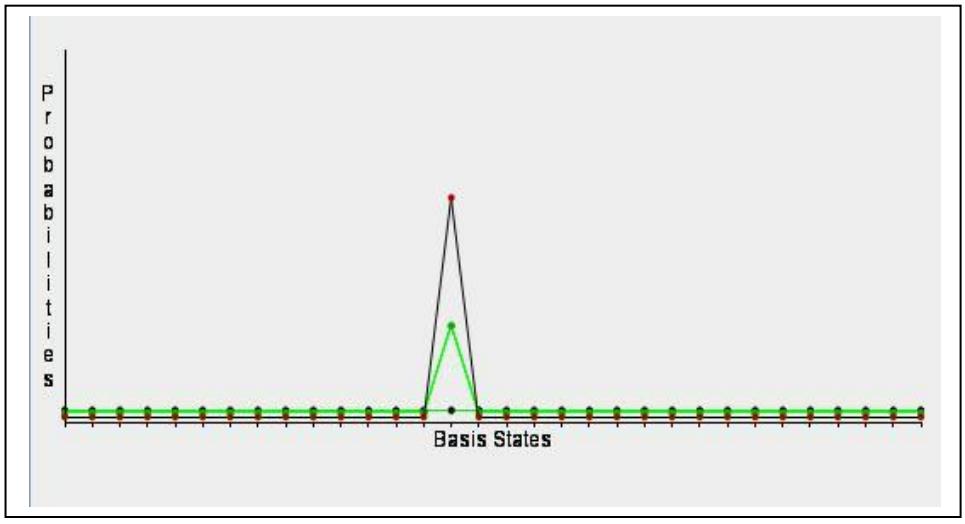


Figure 17: Probability Distributions

This view provides a look at the algorithm's probability distribution as each point represents a different basis state. Since we initialized 5 qubits we see 32 or 2^5 values displayed in this animation sorting four integers. As stated above, recall that the amplitudes start out with the same value 1 divided by the square root of the number of basis states or $1/\sqrt{32}$ in this initial setup and the probability of being in state $|1\rangle$ is $|amp^2|$ so the probabilities are initialized to 0.031249 at the start of the algorithm's execution. As the phase inversion and inversion about the mean's operations continue and new values are calculated the amplitude of the marked basis state will be amplified and the probability of that state will also increase as shown in the

screenshot above. Different colors portray the changes as the algorithm loops through the operations and the probability spikes. The animator is limited to sorting four integers at this point as sorting more would require 5! = 120 or 6! = 720 values needing 2^7 or 128 and 2^{10} or 1024 qubits respectively and would require more space allotted for this view or smaller visuals.

The amplitude view provides a look at Grover's inversion about the mean as it provides the user with a demonstration of the different phases of that operation. The first animation is shown here:

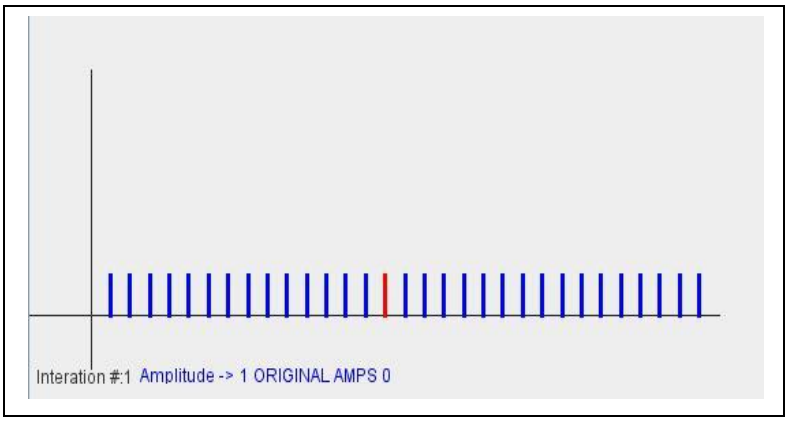


Figure 18: Initial Amplitudes

This animation has all the amplitudes initialized to equal values with the red bar indicating the marked amplitude. In a conventional computer the state of the machine is a single string of ones and zeros, but the state of a quantum computer is a vector with components for every possible string of ones and zeros. These strings of ones and zeros form the basis for a vector space and in this first amplitude animation, this machine state is captured by QuAL with the following vector space:

: Initial Amps:
: STATE:
0.17678 |0> + 0.17678 |1> + 0.17678 |2> + 0.17678 |3> + 0.17678 |4> + 0.17678 |5>

+ 0.17678 |6> + 0.17678 |7> + 0.17678 |8> + 0.17678 |9> + 0.17678 |10> + 0.17678

|11> + 0.17678 |12> + 0.17678 |13> + 0.17678 |14> + 0.17678 |15> + 0.17678 |16> +

0.17678 |17> + 0.17678 |18> + 0.17678 |19> + 0.17678 |20> + 0.17678 |21> +

0.17678 |22> + 0.17678 |23> + 0.17678 |24> + 0.17678 |25> + 0.17678 |26> +

0.17678 |27> + 0.17678 |28> + 0.17678 |29> + 0.17678 |30> + 0.17678 |31>

Figure 19: Initial Vector State

The next animation of the amplitudes exhibits the negation phase where *CPhase( )* has flipped the marked amplitude shown in Figure 20 with Figure 21 containing the represented data:

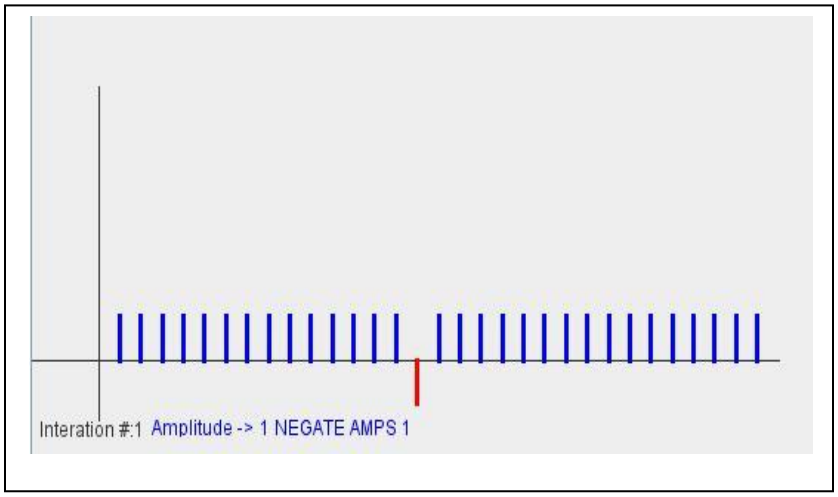


Figure 20: Negation Phase

: Negate Amps:
: STATE:
0.17678 |0> + 0.17678 |1> + 0.17678 |2> + 0.17678 |3> + 0.17678 |4> + 0.17678 |5>

+ 0.17678 |6> + 0.17678 |7> + 0.17678 |8> + 0.17678 |9> + 0.17678 |10> + 0.17678

|11> + 0.17678 |12> + 0.17678 |13> **- 0.17678 |14>** + 0.17678 |15> + 0.17678 |16> +

0.17678 |17> + 0.17678 |18> + 0.17678 |19> + 0.17678 |20> + 0.17678 |21> +

0.17678 |22> + 0.17678 |23> + 0.17678 |24> + 0.17678 |25> + 0.17678 |26> +

0.17678 |27> + 0.17678 |28> + 0.17678 |29> + 0.17678 |30> + 0.17678 |31>

Figure 21: Negate Phase Data

Figures 22 - 29 exhibit the animation as the algorithm calculates the average of the amplitudes and flips the marked amplitude using the formulas defined by Grover's algorithm. The series is animated by QuAL and runs for three iterations before repeating the entire procedure. The corresponding probabilities run concurrently exhibiting the elevation of the 14th state with every iteration of the loop.

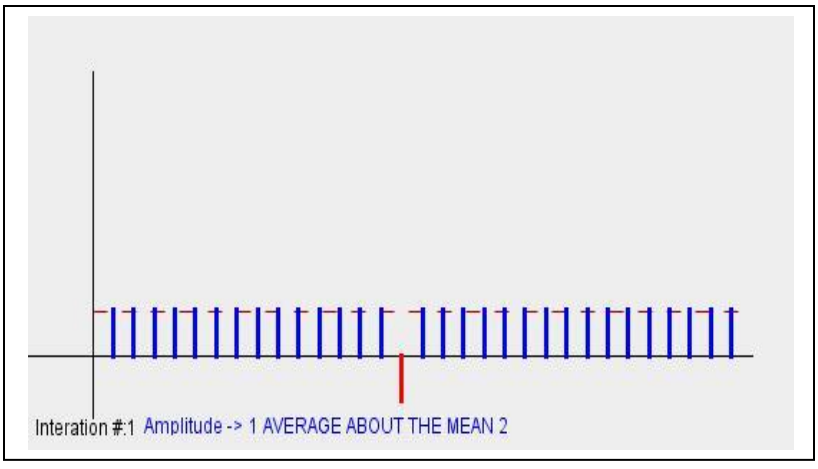


Figure 22: Calculating the Average

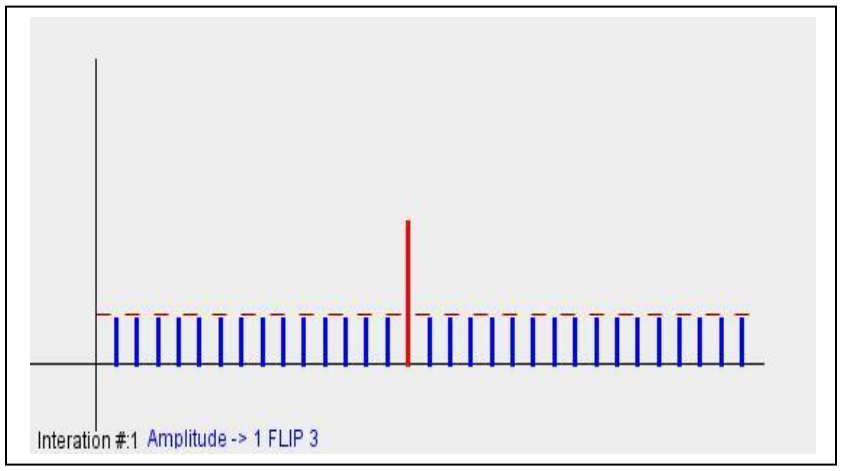


Figure 23: Amplification

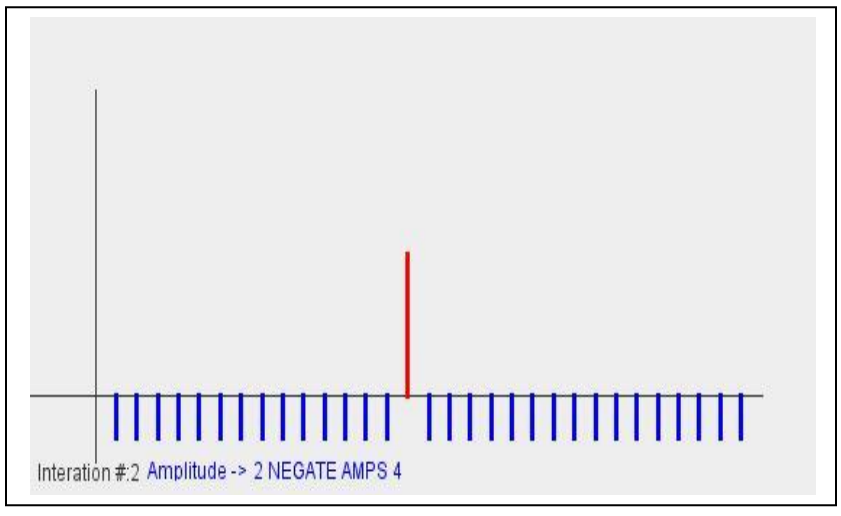


Figure 24: Repeat Negate

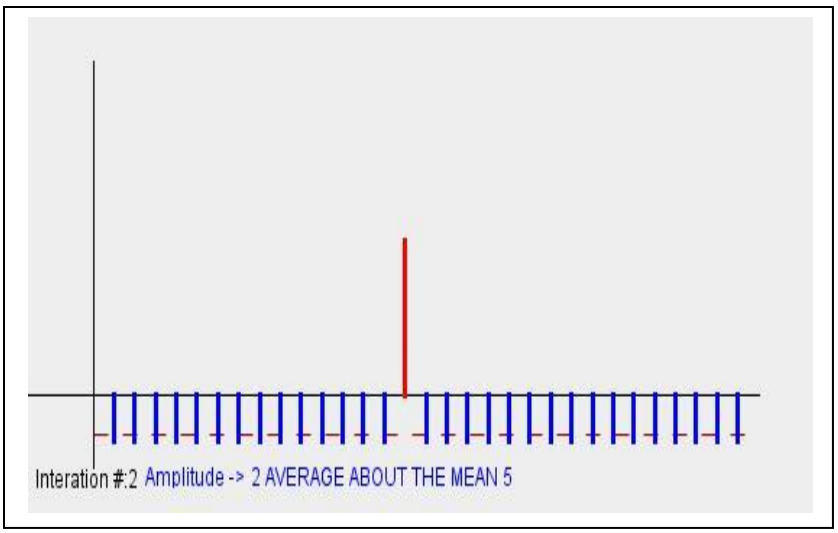


Figure 25: Repeat Average

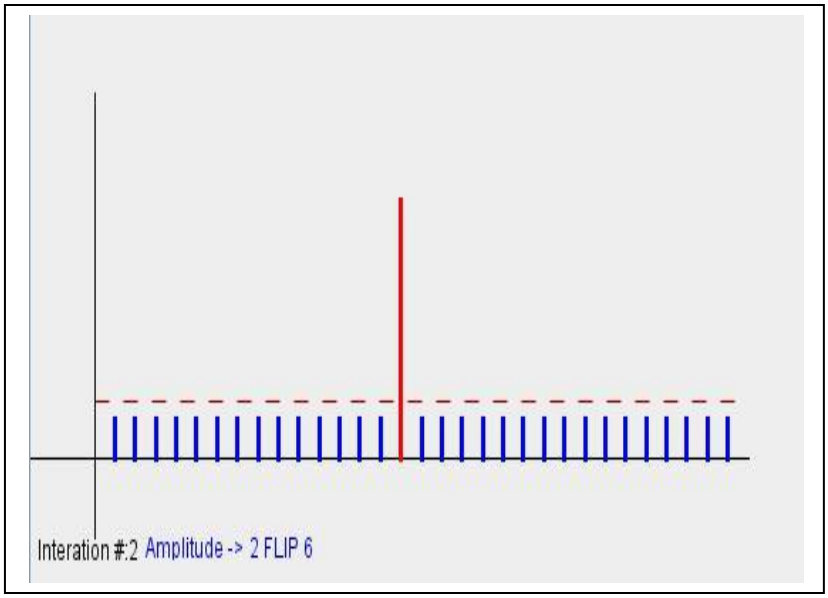


Figure 26: Repeat Amplify

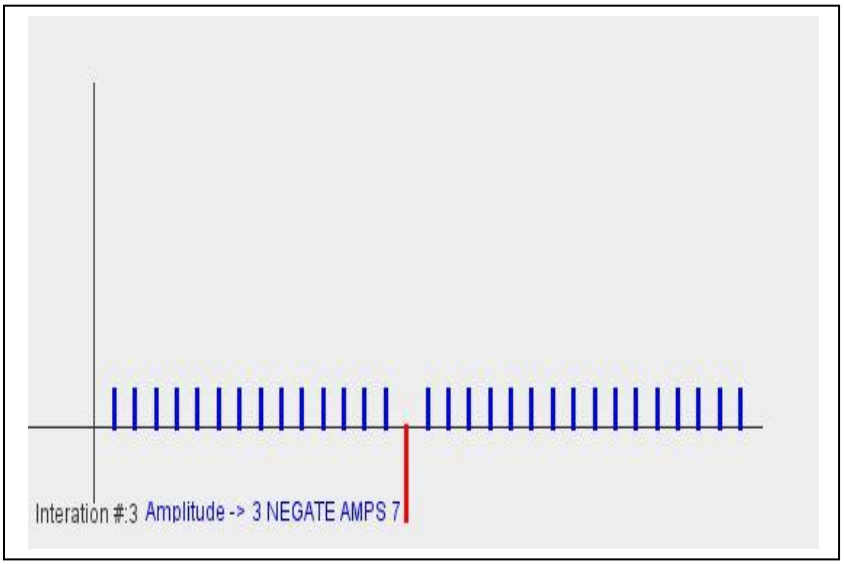


Figure 27: Final Negate

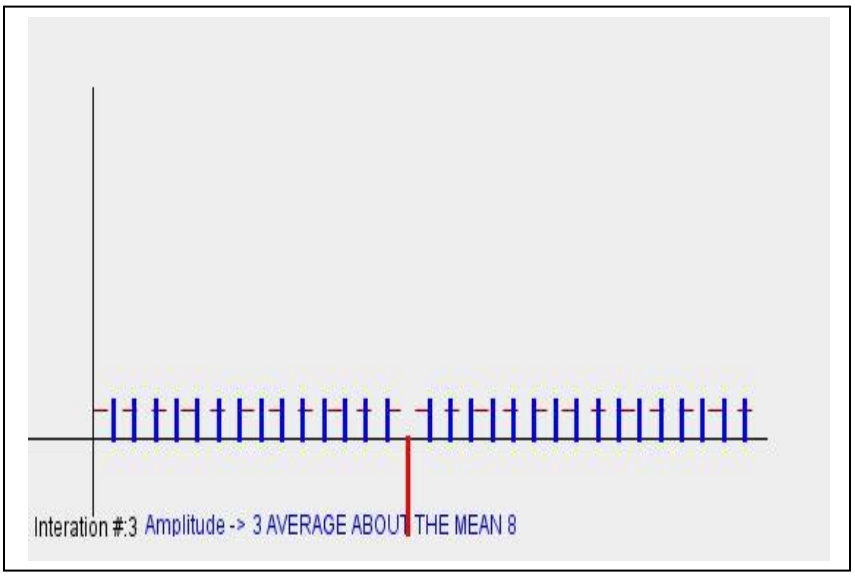


Figure 28: Final Average

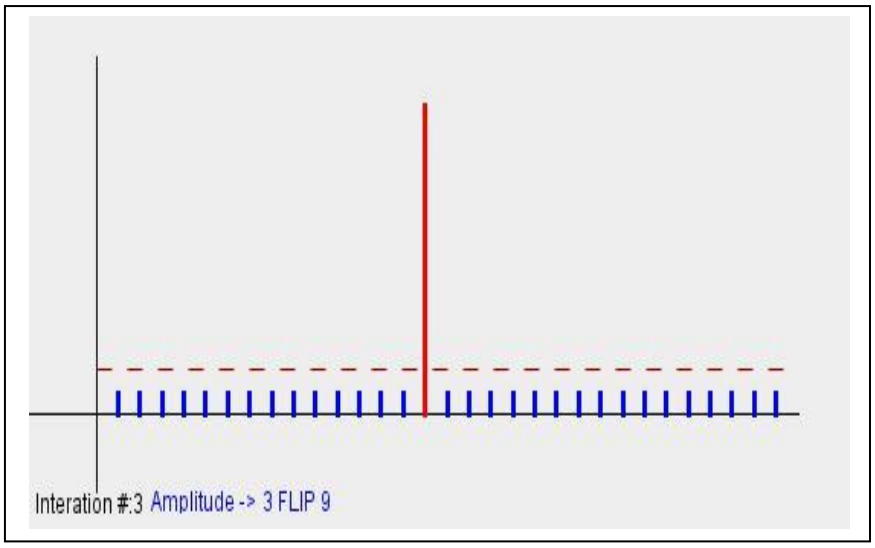


Figure 29: Final Amplification

The final components of QuAL provide a textual representation of important concepts enhancing the two animation views and code highlighting. The input box is non-editable and displays the initial set {6,2,1,9} representing the integers in an unsorted permutation and the speed control provides the ability to change the speed of all the views.

Case Study Results

The main goal of this research was to develop a quantum algorithm animator. A secondary goal was to use it in an educational setting with computer science students to investigate its usefulness in learning about quantum concepts and to collect feedback about its interface and animations. QuAL was used by 11 computer science students in a case study that took place at Wayne State College in Wayne, NE. Twenty students participated in the study.

Common computer science teaching instruments will typically include lectures with or without visuals, textbooks, and lab exercises or experiments. Algorithm animators have been added to this list of tools and have become an effective means of presenting visualizations of algorithmic concepts to students (Stasko, 1997). Badre et al. (1991) performed a survey of

computer science instructors and found out that 81% of these instructors use at least one of these methods to teach algorithm principles to students. Urquiza-Fuentes and Velazquez-Iturbide (2009) in their survey of successful algorithm animation systems, discovered in their experiments that viewing animations can improve knowledge acquisition but animation systems should include additional text or narrative contents.

This study used QuAL in an educational setting to validate the hypothesis that visualizations are beneficial pedagogical tools when using them to teach quantum algorithm concepts. The hypothesis was that students would learn more effectively using QuAL than from simply listening to a lecture, as indicated by their performance in pre-tests and post-tests. This research compared student performance after they interacted with QuAL and listened to a quantum computing lecture that included some visualizations but no animation. Two groups were formed after the lecture material was presented to the twenty participating students with Group A taking the post-test, exit survey only, and Group B interacting with QuAL and then taking the post-test and exit survey.

The animator was developed as a Java applet, which is a special kind of Java program that a browser enabled with the Java Plug-in software can run. This method allowed students to access QuAL from any location with Internet access. The case study was presented to students in the Computer Technology and Information Systems (CTIS) lab located at Wayne State College and accessed QuAL's code residing on the CTIS server. This method would provide beneficial results for any educational setting to allow portability and access from any computer running any operating system. The lab computers are running Microsoft Windows XP and the subjects were allowed to use any browser of their choice. Firefox, Internet Explorer, and Chrome are the browsers installed on all of the lab computers. To access QuAL during the case study, Group B

was instructed to open a browser and type in the following URL:

http://bst-lab-net.wsc.edu:280/~lnichols/QuAL

Appendix C contains the data recorded from grading the pre-tests taken by all twenty students. The subjects were given one ID number when they entered the lab, labeling all their material ranging from 831001 – 831020. The subject ID number is in the first column with the question numbers in subsequent columns. The number one was placed in the row when a subject answered a question correctly with the total number correct at the end of each row. The total number at the end of each column contains the correct answers for each question. Figure 30 provides an overview of the correct answers:

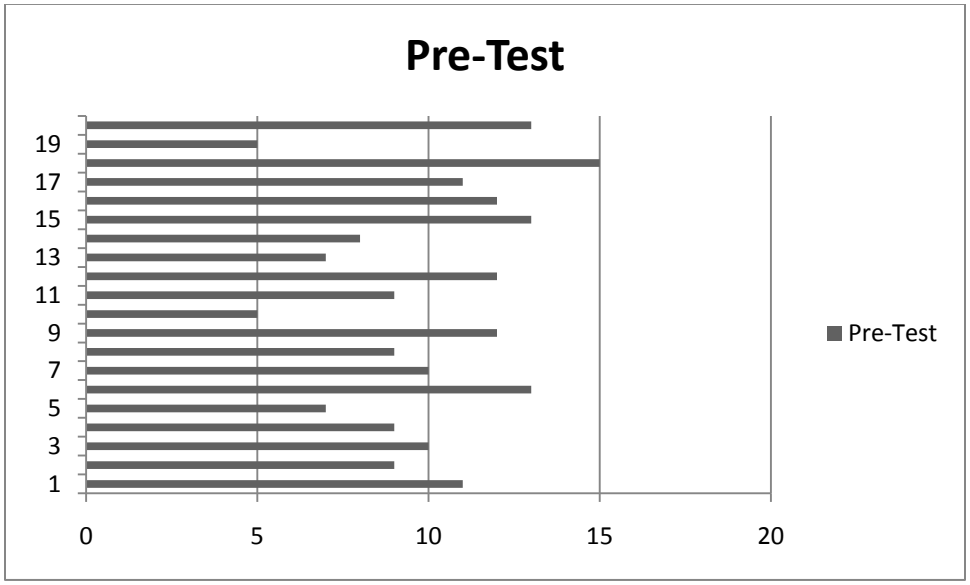


Figure 30: Pre-Test Correct Answers

Group A and Group B results divide the post-test data. Appendix D provides an overview of both Group A and B post-test results. Figure 31 displays an overview of the post-test correct answers:

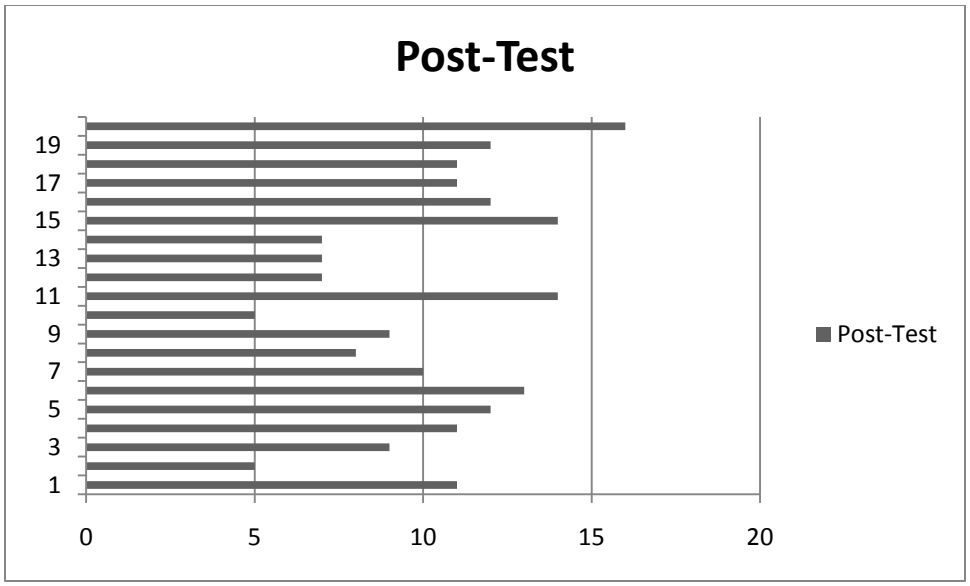


Figure 31: Post-Test Correct Answers

Figure 32 displays Group A's pre-test / post- test comparison scores:

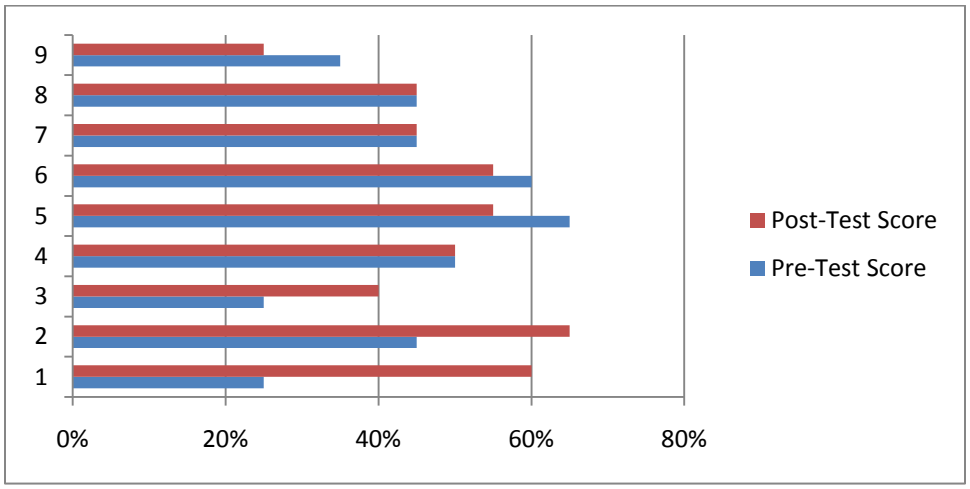


Figure 32: Group (A) Pre-Test / Post Test Comparisons

Figure 33 displays Group B's pre-test / post-test comparison scores:

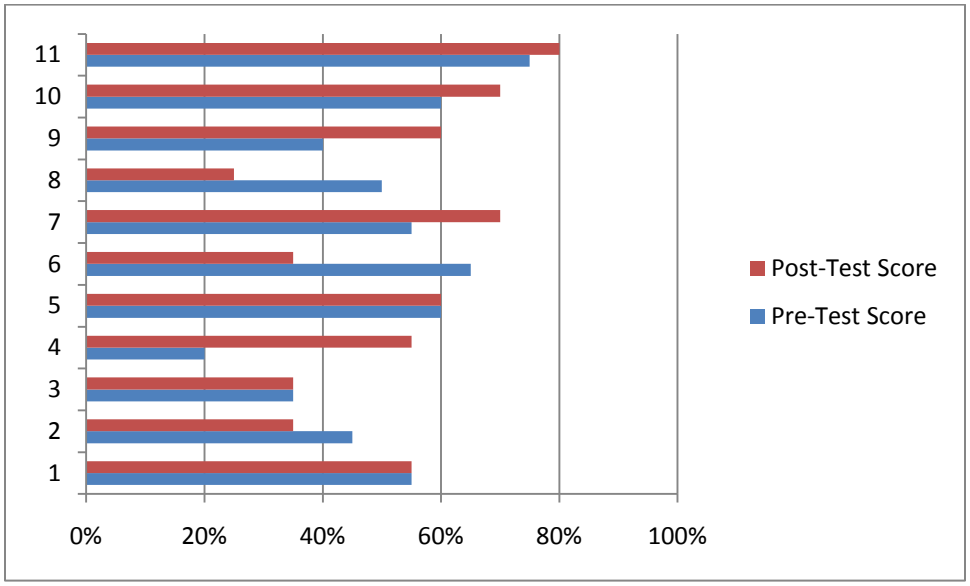


Figure 33: Group (B) Pre-Test / Post Test Comparisons

Data tables for both Group A and Group B's pre-test and post-test scores are shown in the following Tables 2 and 3:

QuAL Case Study			
Group A	Pre-Test Score	Post-Test Score	Improvement
831001	25%	60%	35%
831002	45%	65%	20%
831007	25%	40%	15%
831008	50%	50%	0%
831010	65%	55%	-10%
831012	60%	55%	-5%
831014	45%	45%	0%
831016	45%	45%	0%
831018	35%	25%	-10%
Mean	44%	49%	5%
Median	45%	50%	0%

Table 2: Group (A) Pre-Test / Post-Test Data

QuAL Case Study			
Group B	Pre-Test Score	Post-Test Score	Improvement
831003	55%	55%	0%
831004	45%	35%	-10%
831005	35%	35%	0%
831006	20%	55%	35%
831009	60%	60%	0%
831011	65%	35%	-30%
831013	55%	70%	15%
831015	50%	25%	-25%
831017	40%	60%	20%
831019	60%	70%	10%
831020	75%	80%	5%
Mean	51%	53%	2%
Median	55%	55%	0%

Table 3: Group (B) Pre-Test / Post-Test Data

Examining the preliminary data shows a slight increase in the mean post-test score taken by Group B as opposed to the mean post-test score of Group A, with only a 2% improvement in pre-test versus post-test scores. Group A results show a 5% improvement in pre-test versus post-test scores. Further examination using the dependent *t-test* statistical process produces the following results:

QuAL Case Study *t-test*: Two-Sample Assuming Equal Variances		
	Group A Post-Test	Group B Post-Test
Mean	9.777777778	10.54545455
Variance	5.694444444	12.67272727
Observations	9	11
Pooled Variance	9.571268238	
Hypothesized Mean Difference	0	
df	18	
t Stat	-0.552072677	
P(T<=t) one-tail	0.293844216	
t Critical one-tail	1.734063592	
P(T<=t) two-tail	0.587688432	
t Critical two-tail	2.100922037	

Table 4: *t-Test* Results

Analysis of the *t-test* data using the following steps provides a similar assumption. The null hypothesis asserts that Group A post-test scores will be equal to or show no difference when compared to Group B post-test scores or $H_0 : \mu_1 = \mu_2$. The alternative hypothesis asserts that Group A post-test scores will be lower than Group B post-test scores or $H_1 : \mu_1 < \mu_2$. The above data states that *t*= -0.552, *df*=18 so we could reject H_0 if the value of *t* is <= -1.734. The *t* value is not less than the *t* Critical one-tail value therefore we cannot reject our null hypothesis suggesting that there is not sufficient enough evidence to claim that Group B benefited more by using QuAL. The mean post-test scores are slightly higher so some impact may have occurred but a significant improvement was not found in this study.

A comparison of exam questions looking for a particular concept improvement provided the following data shown in Table 5:

QuAL	Correct Answers			
Case Study	Pre-Test	Group A Post-Test	Group B Post-Test	Total Two Groups
1	7	2	6	8
2	12	8	9	17
3	14	4	2	6
4	8	6	7	13
5	7	3	8	11
6	11	6	8	14
7	14	7	3	10
8	11	8	9	17
9	11	4	6	10
10	14	6	9	15
11	16	3	4	7
12	8	4	6	10
13	12	5	7	12
14	15	4	8	12
15	7	2	2	4
16	5	4	3	7
17	12	2	6	8
18	5	2	2	4
19	4	6	9	15
20	7	2	2	4

Table 5: Question Group Comparisons

This data was used to compare the sections defined in the case study design and shown here in Table 6:

QuAL	Case Study		Totals	
			Pre-Test	Post-Test
Section A	Basic QC Knowledge	1,5,9,13,17	49	49
Section B	Quantum States	2,6,10,14,18	57	62
Section C	Classical / Quantum Sorting	3,7,11,15,19	55	42
Section D	Quantum Algorithms	4,8,12,16,20	39	51

Table 6: Section Results

The above data suggests that basic quantum computing knowledge was common to the students before the listening to the lecture or using QuAL but participation in the case study improved their knowledge of quantum states as well as quantum algorithms. This data was collected to provide feedback for potential improvement areas in future versions of QuAL focusing on the conceptual lessons rather than the aesthetics of its features. Figure 34 provides a visual representation of the data from Table 5:

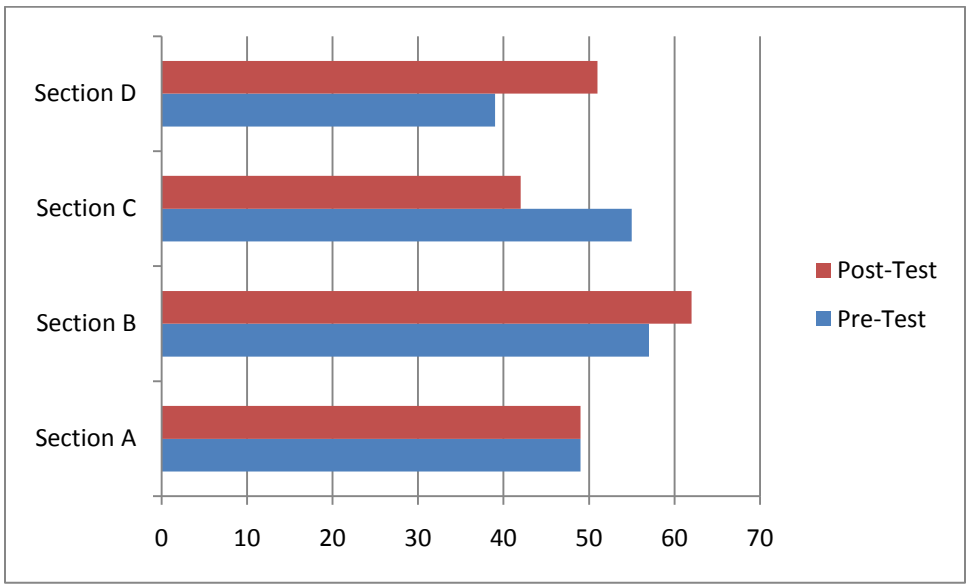


Figure 34: Pre-Test / Post-Test Question Comparisons

Figure 35 provides a comparison of post-test results between Group A and Group B students.

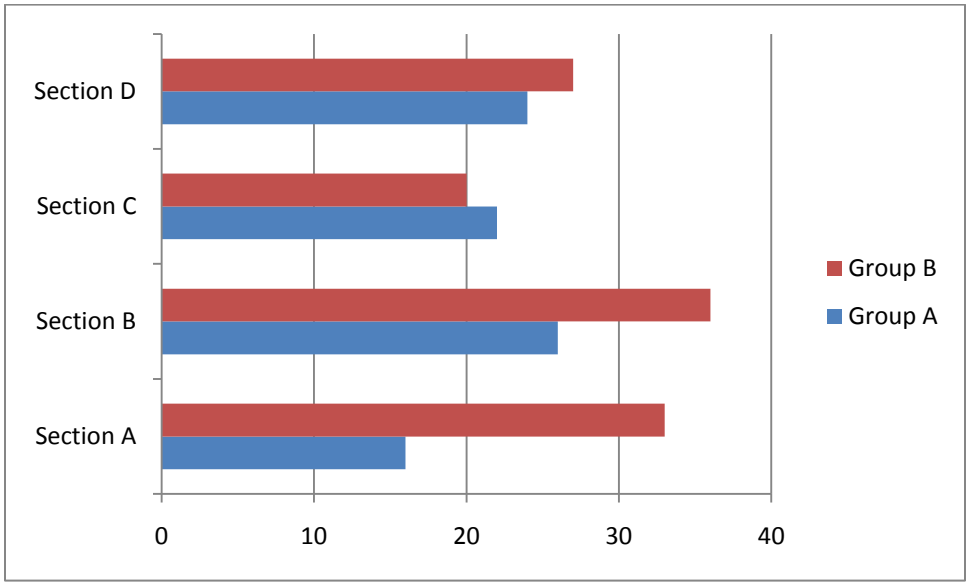


Figure 35: Group A / B Question Comparisons

The final data collection gathered during this case study was acquired by asking the students to complete an exit survey. The exit survey given to the subjects can be found in Appendix F with the data results found in Appendix E. This survey asked questions concerning the material presented during the lecture and about the use of QuAL as well as asking the user to type in additional comments. Most users found the lecture material very challenging but found it increased their interesting in quantum computing. Group B answered the questions concerning the use of QuAL with 72% finding the interface easy to use, 92% found the features useful for understanding quantum algorithms, and 100% of the users thought it supplemented the lecture material. Other interesting findings included 90% of the users thought that the animator increased their basic algorithmic knowledge and 82% agreed that it increased their interest in quantum computing. Some of the consistent comments made by users of QuAL were that Section C (the view that animated the amplitudes) could display longer. Several thought it blinked too quickly from view not giving them enough time to concentrate on its principles.

Others thought the animator was helpful and would like to spend more time using it and about quantum computing concepts.

Summary of Results

The research goal of developing a quantum algorithm animator was actualized by the development of QuAL as described in this section. Java classes interacted with the quantum simulator QCL to animate the algorithm's operations using a data-driven approach. Wallace and Narayanan's quantum sorting algorithm promoted the concepts of quantum computing during execution captured and displayed by QuAL. Probability distributions and amplitudinal changes were mapped to views that provided a graphical representation of the interesting data structures of the algorithmic code. The sorting quantum algorithm's state changes were reflected in the graphical interpretations of QuAL's code as it ensured consistent communication by all views including the highlighted code view. Although this research used many of the exploratory study's findings of successful features and operations, QuAL is still only an initial prototype of a quantum algorithm animator pedagogical tool. Interface concerns, adding features and student comments after usage will be a topic for future research.

To summarize the overall case study findings, small gains were found in the mean scores of Group B's post-test results but t-test results found them to be insufficient enough to claim without a doubt that QuAL assisted with quantum computer learning. There were key findings within quantum concept learning but several perplexing concerns emerged from the final tabulated results. Several students did well in pre-test scores but dropped drastically in post-test scores in both groups. All materials used by this case study as well as some of QuAL's code can be found in the Appendix section of this final report.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

The primary goal of this research was to develop a quantum algorithm animator. A secondary goal to use it in an educational setting was realized to gain information about how to improve it and how effectively it could be used to teach quantum concepts to computer science students. Furthermore, a hypothesis was proposed based on the success found by classical algorithm animators, that QuAL would improve quantum concept learning when used as a pedagogical tool along with supplementary material. The three stages of this research were undertaken to gather information about classical algorithm animators and quantum concepts, use this information to design and develop a quantum algorithm animator, and then use QuAL as a pedagogical tool.

The sorting quantum algorithm used by QuAL and developed by Wallace and Narayanan (2001), provided an elegant solution to the sorting problem using a derivative of Grover's (1998) foundational quantum algorithm. Grover's algorithm provides an efficient solution to database searching and Wallace and Narayanan incorporate these principles into their sorting algorithm altering a search solution into a search then find the correctly sorted permutation solution. Quantum algorithms can be challenging to learn but comparing their concepts to classical concepts assist student learning. The sort problem provided that connection between the quantum realm and classical realm in learning quantum computing principles.

The case study statistics show that there are no differences in performance on the post-tests between subject groups A or B. Thus, the addition of a quantum algorithm animator offers no

significant performance gains over presenting only a quantum computing lecture. The research hypothesis had predicted gains in the group using the animator, but the *t-test* stats showed a different interpretation. The null hypothesis could not be rejected therefore no appreciable difference was measured in the post-tests of the two subject groups. This does not mean that the animator group did not perform better as the mean values claim, but statistically the research cannot claim a significant difference.

A reason for the low gains from the animator groups may have been the difficulty of the test. Examining the pre / post-test improvement results show six students achieving lower scores on post-tests than on the pre-tests from both groups. Did they just guess at answers on the pre-test? Did they doubt their answers on the pre-test and 'over-think' answers on the post-test or did using the animator present the material differently? None of the members of either groups taking pre or post-tests attained a perfect score. A member of the animator group scoring 80% on the post-test achieved the best score and four out of the top five scores were achieved by members of the animator group. The mean differences between the pre and post-test by both groups was less than 5% so, on average, neither group had a significant change in test scores after listening to the lecture and / or interacting with the animator.

Another possible reason for low gains by the animator group is that the test may not have been a good indicator of how well students understood quantum concepts. The concepts are challenging as many of the students noted in their comments and they are new concepts to many computer science students. The students may have heard about the quantum concepts but none of the students in the case study had ever researched the topic beyond basic curiosity.

An interesting finding about individual quantum concepts emerged from the data collected concerning those five groups:

1. Section A: Basic QC Knowledge
2. Section B: Quantum States
3. Section C: Quantum Sorting
4. Section D: Quantum Algorithms

Basic quantum knowledge did not seem to increase but remained the same from pre-test to post-test results. Quantum states and classical / quantum sorting had a mild increase and decrease but questions relating to quantum algorithms increased by 13%.

Student comments concerning QuAL were positive and constructive in nature. A prevailing comment made by several students asked for a reduction in animation timing between views of the quantum algorithms amplitudes. Many claimed the animator increased their interest in the quantum computing field and stated that they would continue their own personal research. The interface aesthetics did not receive any student comments. The animator seemed to operate appropriately except for the timing issue mentioned in the comment section of the exit survey.

Implications and Recommendations

QuAL is in need of some modifications before using it as a pedagogical tool in a computer science course but this initial design has fulfilled the goal of this research. The timing issues could be implemented quickly and the class structure is setup for the addition of new quantum algorithms but additional testing should be used to determine whether the pre / post-tests, the student tutorial, or the lecture material need to be adjusted. The post-test scores could have been improved by modifying and coordinating the material or incorporating the use of QuAL into a course where more preparation time could be dedicated to its use.

The case study could have been modified to run over two days instead of one day. The first day could have been dedicated to presenting the lecture material and then have one group return

on a second day to interact with the animator. Perhaps the timeframe and the material presented lost some validity as student retention dropped after a long day of work or school. This could explain the decrease in pre-test versus post-test scores recorded by both groups. The following is a list of suggested changes gathered during this research:

1. Slow change time in amplitude animations.
2. Alternate the probability distribution animations enhancing changes in data.
3. Add more algorithms and different illustrations to provide another look at the quantum concepts.
4. Add step button and backtracking capabilities.
5. Allow user input to change sorted integers.

Summary

To summarize the overall findings of this research, the gains found by using QuAL in an educational setting were slight but QuAL did provide a small advantage over just listening to the lecture material. QuAL's performance during the case study was acceptable as the Java applet executed its operations as coded without any errors or issues. It is difficult to design effective animations especially when attempting to illustrate quantum principles and further research in refining these designs and concepts and testing them through empirical studies of quantum algorithm animation is recommended. Longer or more extensive studies may offer better statistical outcomes. This research was interested in the fundamental question of whether animations can assist students learn and understand quantum algorithms. Although case studies such as the one performed by this research cannot answer that fundamental question, they can supply future research with data on how students interact with different animations and provide potential learning strategies used by students.

Finally, attempting to illustrate concepts from a designer's perspective may not be conveyed properly through the external representation of the animations to map into the viewers' internal representation. In other words, quantum computing is a complex topic and there is no simple means of designing effective visualizations. A main objective described by this research was to entice students into wanting to learn more about quantum computing and QuAL has successfully accomplished this task as shown by student comments received in the case study's exit survey.

Appendix A
Wayne State College IRB Approval

Wayne State College
Internal Review Board
1111 Main Street
Wayne, NE 68787
May 26, 2010

Lori Nicholson
IRB Proposal #49
Wayne State College
CTIS Department
Gardner Hall

Dear Lori:

I have reviewed your research proposal and find that it meets the criteria for exempt status.

Good luck on your research. It sounds like an interesting study.

Sincerely,

JoAnn Bondhus
WSC IRB Committee Member

Appendix B
Nova Southeastern University IRB Approval

To: Lori Nicholson
From: Ling Wing, Ph.D.
Internal Review Board
Date: July 20, 2010
Re: Quantum Algorithm Animator
IRB Approval Number: wang06151001

I have reviewed the above-referenced research protocol at the center level. Based on the information provided, I have determined that this study is exempt from further IRB review. You may proceed with your study as described to the IRB. As principal investigator, you must adhere to the following requirements:

1) CONSENT: If recruitment procedures include consent forms these must be obtained in such a manner that they are clearly understood by the subjects and the process affords subjects the opportunity to ask questions, obtain detailed answers from those directly involved in the research, and have sufficient time to consider their participation after they have been provided this information. The subjects must be given a copy of the signed consent document, and a copy must be placed in a secure file separate from de-identified participant information. Record of informed consent must be retained for a minimum of three years from the conclusion of the study.

2) ADVERSE REACTIONS: The principal investigator is required to notify the IRB chair and me (954-262-5369 and 954-262-2020 respectively) of any adverse reactions or unanticipated events that may develop as a result of this study. Reactions or events may include, but are not limited to, injury, depression as a result of participation in the study, life-threatening situation, death, or loss of confidentiality/anonymity of subject. Approval may be withdrawn if the problem is serious.

3) AMENDMENTS: Any changes in the study (e.g., procedures, number or types of subjects, consent forms, investigators, etc.) must be approved by the IRB prior to implementation. Please be advised that changes in a study may require further review depending on the nature of the change. Please contact me with any questions regarding amendments or changes to your study.

The NSU IRB is in compliance with the requirements for the protection of human subjects prescribed in Part 46 of Title 45 of the Code of Federal Regulations (45 CFR 46) revised June 18, 1991.

Appendix C
Case Study Overall Pre / Post Test Results

QuAL Case Study Results Pre-Test																					Correct Answers
Q-No's	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
831003		1	1	1	1				1	1	1	1		1	1		1				11
831004	1	1					1			1	1		1	1			1			1	9
831008			1	1			1	1		1	1		1	1	1		1				10
831014				1		1	1	1	1		1					1			1	1	9
831005										1		1	1	1	1	1		1			7
831009		1	1			1	1	1	1	1	1		1	1	1		1			1	13
831015		1				1	1	1	1	1	1	1		1						1	10
831002			1		1	1	1	1		1		1			1		1				9
831006	1	1	1	1	1	1		1	1	1	1		1				1				12
831007			1										1	1		1	1				5
831016			1	1			1	1		1	1	1				1		1			9
831012		1	1			1	1		1	1	1	1	1	1			1	1			12
831018	1		1				1			1	1	1			1						7
831017		1	1					1	1	1	1	1		1							8
831010	1	1	1		1	1	1		1				1	1		1	1		1	1	13
831019	1	1	1	1		1	1	1	1	1	1		1	1							12
831013	1	1			1	1		1	1		1		1	1			1		1		11
831020	1	1		1	1	1	1	1	1		1		1	1	1		1	1		1	15
831001			1		1		1				1			1							5
831011		1	1	1		1	1			1	1		1	1			1	1	1	1	13
Totals Correct Per Question	7	12	14	8	7	11	14	11	11	14	16	8	12	15	7	5	12	5	4	7	

Appendix D
Case Study Post-Test Results

QuAL Case Study Results																					Correct
Post-Test																					Answers
Q-No's	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Group A																					
831010		1	1	1		1	1	1		1				1		1			1	1	11
831018							1	1				1					1		1		5
831014		1		1			1	1			1	1			1	1			1		9
831012		1			1	1		1	1	1	1	1	1			1			1		11
831001	1	1	1	1		1	1	1	1	1		1						1	1		12
831002	1	1	1	1	1	1	1	1					1	1		1	1			1	13
831008		1		1		1		1	1	1	1		1	1	1						10
831007		1	1		1		1			1			1					1	1		8
831016		1		1		1	1	1	1	1			1	1							9
Total Correct Per Question	2	8	4	6	3	6	7	8	4	6	3	4	5	4	2	4	2	2	6	2	
Group B																					
831015							1	1	1										1	1	5
831013	1	1		1	1	1		1	1	1	1	1	1				1	1	1		14
831004	1									1	1		1	1		1			1		7
831005	1	1			1					1			1	1					1		7
831011		1		1		1		1		1				1		1					7
831019	1	1		1	1	1		1	1	1		1		1			1	1	1	1	14
831017		1	1	1	1	1	1	1				1	1	1			1		1		12
831003		1		1	1	1		1	1	1			1	1		1			1		11
831006	1	1	1		1	1		1	1	1		1					1		1		11
831009		1		1	1	1		1		1	1	1	1	1	1		1				12
831020	1	1		1	1	1	1	1	1	1	1	1	1	1	1		1		1		16
Total Correct Per Question	6	9	2	7	8	8	3	9	6	9	4	6	7	8	2	3	6	2	9	2	

Appendix E
Exit Survey Results and Comments

	QuAL Exit Survey									
	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	SA	A	N	D	SD
1	12	8				60%	40%			
2	17	3				85%	15%			
3	10	7	3			50%	35%	15%		
4	11	7	2			55%	35%	10%		
5	3	6	2			27%	55%	18%		
6	5	3	3			46%	27%	27%		
7	2	4	4	1		18%	36%	36%	10%	
8	2	7	2			18%	64%	18%		
9	6	5				55%	45%			
10			1	6	4			10%	55%	35%
11			1	6	4			10%	55%	35%
12	4	5	2			36%	46%	18%		
13	2	8		1		18%	72%		10%	
14	2	8	1			18%	72%	10%		

Comments:

Analogies used helped me understand the material.

Interesting concepts that challenges the conventional way of thinking and knowledge we have.

Material can be understood with more time.

Would like to see more of the Math taking place.

Animator worked well, however, Region C did not stay visible long enough.

I prefer the images stay up longer to be able to read the explanation as well as view the picture to understand the concepts.

Many new confusing concepts in a short amount of time.

Very confusing at first, but did understand a little of the algorithms at the end.

Section C could display longer.

I wish I had the time to learn about quantum algorithms, but you covered the basics very well.

The lecture increased my interest to research more.

Appendix F
Case Study Exit Survey

Quantum Algorithm Animator

Nova Southeastern University
Dissertation Research
Lori Nicholson

Please answer the following questions as instructed.

Group A – answer questions 1 – 4 (add comments at the end of the survey). Group B – answer all questions (add additional comments at the end of the survey).

Exit Survey ID	
Member of Group (A) or (B)	

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1. The lecture material presented was challenging.					
2. The lecture material reflected research goals.					
3. The lecture material increased interest in quantum computing.					
4. The lecture material provided an organized view of quantum computing.					
Group B Only					
5. The quantum algorithm animator's interface was easy to use.					
6. The time allotted was adequate to understand the quantum algorithm animator's interface.					
7. The time allotted was adequate to understand quantum computing concepts presented in the lecture.					
8. The quantum algorithm's user features were useful for understanding quantum algorithms.					
9. The quantum algorithm animator supplemented the lecture material.					
10. The lecture material did not help me learn about quantum computing.					
11. The quantum algorithm animator did not help me learn about quantum computing.					
12. The quantum algorithm animator increased interest in quantum computing.					
13. The quantum algorithm animator increased basic algorithm knowledge.					
14. The quantum algorithm animator's features were easy to use.					

Appendix G
Case Study Pre / Post Test

Quantum Algorithm Animator

Nova Southeastern University
Dissertation Research
Lori Nicholson

Pre-Post Study Test (Key – correct answers bolded)

Circle : Pre / Post TestID: _____

Please answer the following questions to the best of your abilities:

---

1. In the field of quantum computing, what is superposition?
    a. By measuring some qubits, others automatically reach the desired position
    b. A quantum computer in many states simultaneously
    c. A complex number whose absolute value squared represents a probability

2. How many quantum states can a quantum computer (unmeasured) be in at once?
    a. One
    b. Many
    c. Seven

3. Does quantum sorting provide more efficient solutions to classical sorting?
    a. In some cases
    b. Never
    c. Always

4. Which of the following foundational quantum algorithms could potentially break our current encryption scheme?
    a. Grover's
    b. Shor's
    c. Deutch's

5. In the field of quantum computing, what is entanglement?
    a. A quantum computer in many states simultaneously.
    b. A complex number whose absolute value squared represents a probability.
    c. By measuring some qubits, others automatically reach the desired position.

6. What happens when we measure a quantum computer while it is in superposition?
    a. Nothing
    b. It does not provide an output
    c. It collapses to a single position

7. How can quantum sorting algorithms provide more efficient solutions than their classical analogues?
    a. Wallace and Narayanan's solution was to exploit superposition permutation-based representations for list searching
    b. Shi uses the decision tree model to propose a more efficient solution
    c. Both a and b are correct

8. A quantum computer using superposition that can search all entries of an unordered array simultaneously and find the object in √n queries is an example of which of the following foundational quantum algorithms?
    a. Shor's
    b. Grover's
    c. Deutch's

9. In the field of quantum computing, what is the probability amplitude?
    a. A quantum computer in many states simultaneously.
    b. A complex number whose absolute value squared represents a probability.
    c. By measuring some qubits, others automatically reach the desired position.

10. The input / output of a quantum computer is the same as a classical computer, the difference is during computation and is said to be, at times, a black box.

    a. True

    b. False

11. One of the basic concepts of sorting is to:

    a. Search for a particular item in a listing of items

    b. Place items in some order with two basic operations: compare and swap

    c. Match an item with other items in the list

12. This foundational quantum algorithm demonstrates the power and usefulness of quantum computing for factoring numbers:

    a. Grover's

    b. Deutch's

    c. Shor's

13. What is a qubit?

    a. It is the quantum analogue to the classical bit and the basic unit of quantum information.

    b. It is the basic unit in a classical computer.

    c. It is the quantum analogue to the classical byte.

14. The classical computer will be in either a 0 or 1 state but the quantum qubit can be in:

    a. 0 or 1 or anywhere in between

    b. Only 0 or 1

    c. Only 1

15. A common classical sort algorithm is:

    a. Bubble

    b. Hirschberg's

    c. Hungarian

16. All quantum algorithms work with the following basic framework except:
    a. System will start with the qubits in a particular classical state
    b. A measurement will never take place
    c. System will be put into a superposition of many states

17. The main barrier to the development of a quantum computer is decoherence. What is decoherence?
    a. The loss of purity of the state of a quantum system as the result of entanglement with the environment.
    b. It is when a qubit is in both '0' and '1' states simultaneously.
    c. It is the ability of the quantum system to reach a desired outcome.

18. Any quantum state can be expressed in terms of a sum of:
    a. many complex numbers
    b. only one state
    c. basis states

19. The quantum sorting algorithms you were shown today were commonly based on which foundational quantum algorithm?
    a. Shor's
    b. Deutch's
    c. Grover's

20. Which foundational quantum algorithm solves a slightly contrived problem?
    a. Grover's
    b. Deutsch's
    c. Shor's

Appendix H
Case Study Consent Form

Consent Form for Participation in the Research Study Entitled
Quantum Algorithm Animator

Funding Source: None.

IRB protocol # TBD

Principal investigator
Lori Nicholson
1111 Main St GH 206K
Wayne, NE 68787
(402) 375-7017

Co-investigator
Michael Laszlo, Ph.D.
3301 College Avenue
Fort Lauderdale, FL 33314
(954) 262-2076

For questions/concerns about your research rights, contact:
Human Research Oversight Board (Institutional Review Board or IRB)
Nova Southeastern University
(954) 262-5369/Toll Free: 866-499-0790
IRB@nsu.nova.edu

Or
Institutional Review Board
Wayne State College
402-375-7000

Site Information
Wayne State College
Gardner Hall
Wayne, NE 68787

What is the study about?

You are invited to participate in a research case study. The goal of this study is to test the comparative efficiency of a quantum algorithm animator.

Why are you asking me?

We are inviting you to participate because you are currently enrolled in Wayne State College as a computer science or computer information systems major.

What will I be doing if I agree to be in the study?

You take a pre-study test and then attend a 30-minute lecture. After listening to a lecture you be assigned to either Group A or Group B. Group A students will be asked

to take a 30 minute post-study exam and then fill out an exit survey. Group B students will be asked to interact with a quantum algorithm animator for 30 minutes and then take the post-study exam followed by an exit survey.

Is there any audio or video recording?

This research project will not include any audio or video recordings.

What are the dangers to me?

There are no identifiable risks to participating in this case study.

Are there any benefits to me for taking part in this research study?

You will learn about quantum computing concepts.

Will I get paid for being in the study? Will it cost me anything?

There are no costs to you, you will receive food, and snacks to enjoy while participating in this case study.

How will you keep my information private?

No tests and survey instruments used in this study will require your name or any private information.

What if I do not want to participate or I want to leave the study?

You have the right to leave this study at any time or refuse to participate. If you do decide to leave or you decide not to participate, you will not experience any penalty or loss of services you have a right to receive.

Other Considerations:

If the researchers learn anything which might change your mind about being involved, you will be told of this information.

Voluntary Consent by Participant:

By signing below, you indicate that

- this study has been explained to you
- you have read this document or it has been read to you
- your questions about this research study have been answered
- you have been told that you may ask the researchers any study related questions in the future or contact them in the event of a research-related injury
- you have been told that you may ask Institutional Review Board (IRB) personnel questions about your study rights
- you are entitled to a copy of this form after you have read and signed it
- you voluntarily agree to participate in the study entitled *Quantum Algorithm Animator*

Participant's Signature: _____ Date: _____

Participant's Name: _____ Date: _____

Signature of Person Obtaining Consent: _____

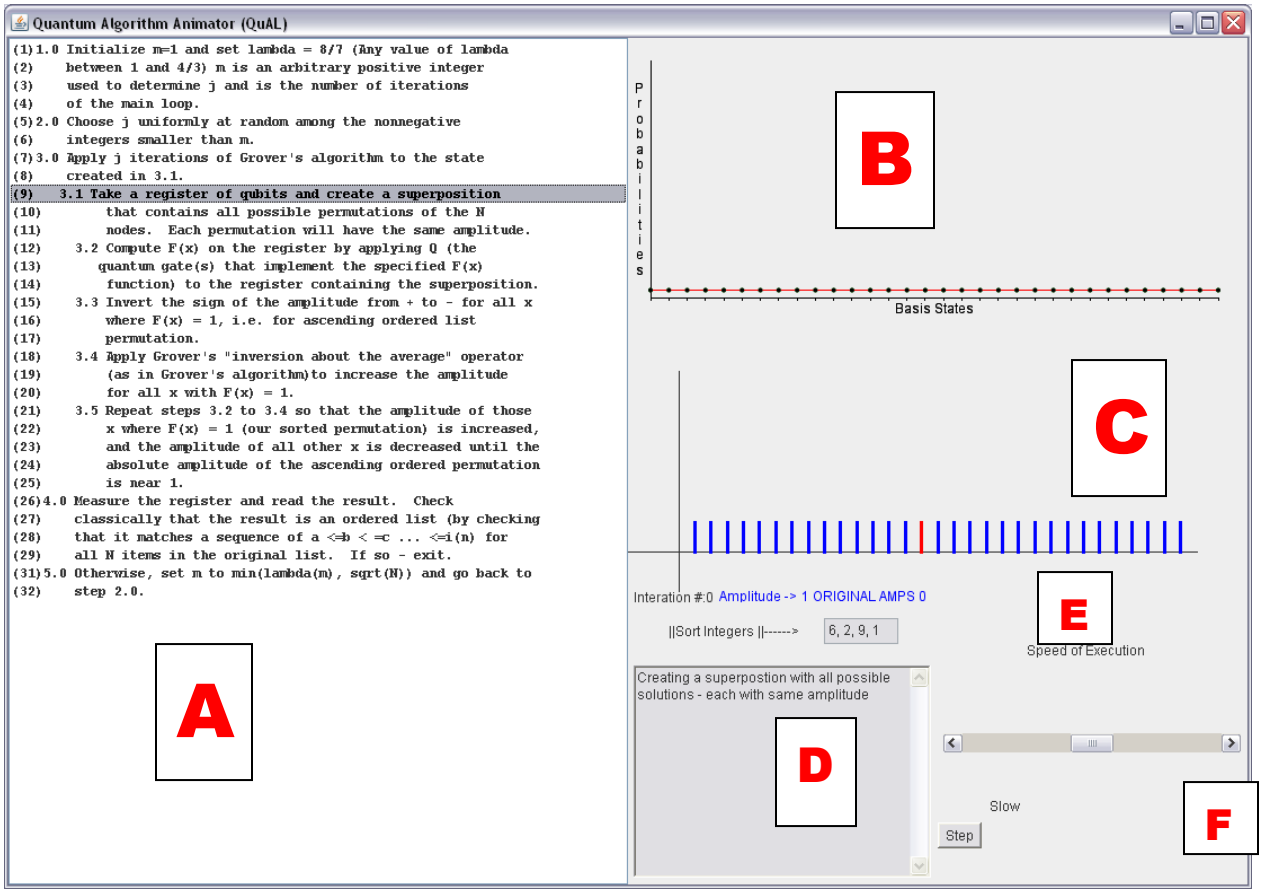
Date: _____

Appendix I
Case Study Student Tutorial - QuAL

Dissertation Case Study using QuAL (Quantum Algorithm Animator)
Lori Nicholson
Nova Southeastern University

Student QuAL Tutorial

1) The Interface



Region A – Is the box containing the algorithm's text. It will highlight the appropriate line of code as the graphics are updated with the excuted code.

Region B – Is the probability of obtaining the correct solution for *f(x)* = 1 or our sorted permutation. The basis states are located on the x-axis with their related probabilites on the y-axis.

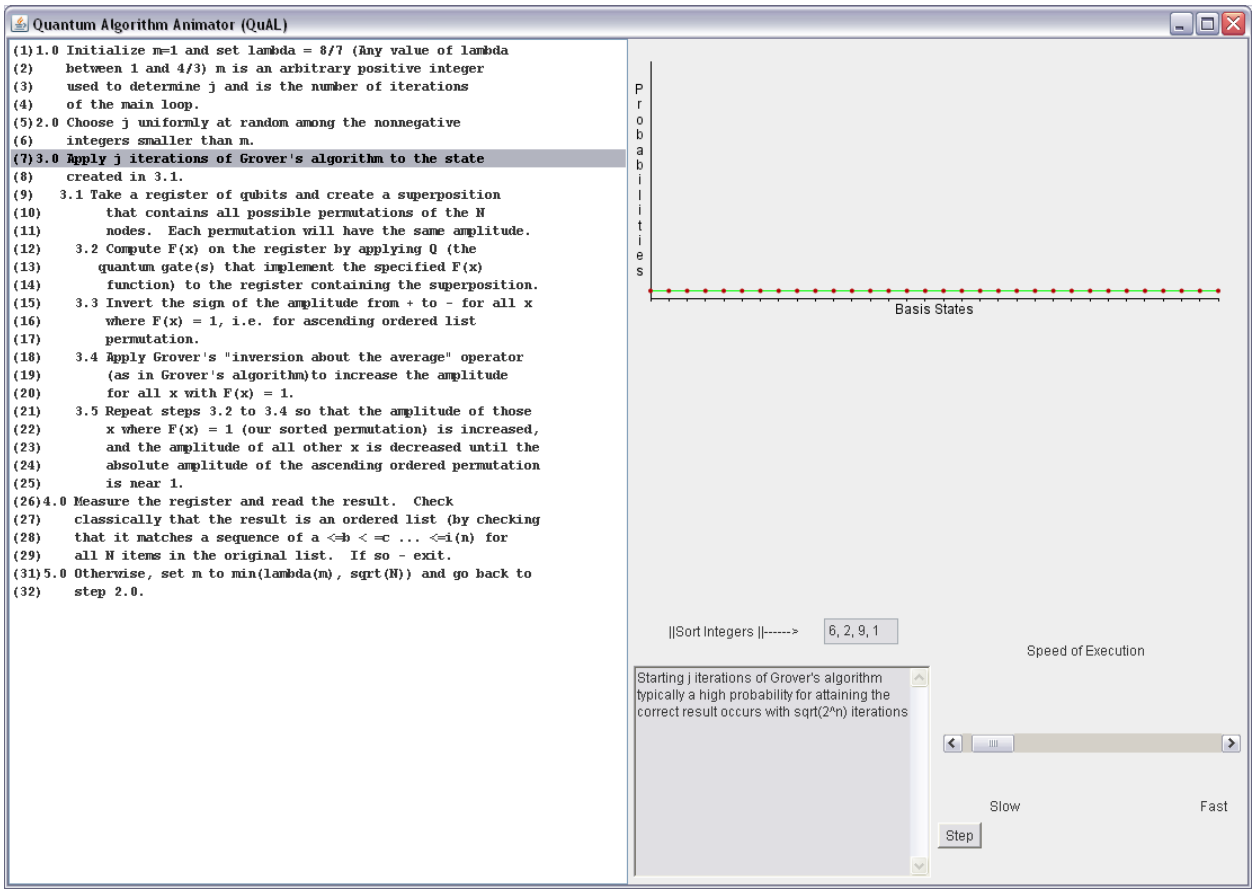
Region C – Contains the graphical representations for the amplitudes. The amplitudes will switch from Original -> Negate -> Average about the Means -> Flip illustrating the different amplitudes found as the quantum algorithm computes these values.

Region D – This text box will provide the user with additional information specific to the highlighted line of text in Region A.

Region E – This region will exhibit the main loop interations, the title for the appropriate amplitude and the sequence of integers being sorted by the quantum algorithm.

Region F – This is the speed control. Moving the slider to the left will slow the animation for easier assessment of the quantum algorithm's operations.

2) A Sample Run of Wallace – Narayanan Quantum Sorting Algorithm (Wallace & Narayanan, 2001)



The animator will start by highlighting the first lines of pseudocode in Section A. These steps initialize all the variables and put the register of qubits into a superposition of all states.

The proposed algorithm exploits the quantum principles of Grover's foundational algorithm using permutation-based representations for list searching.

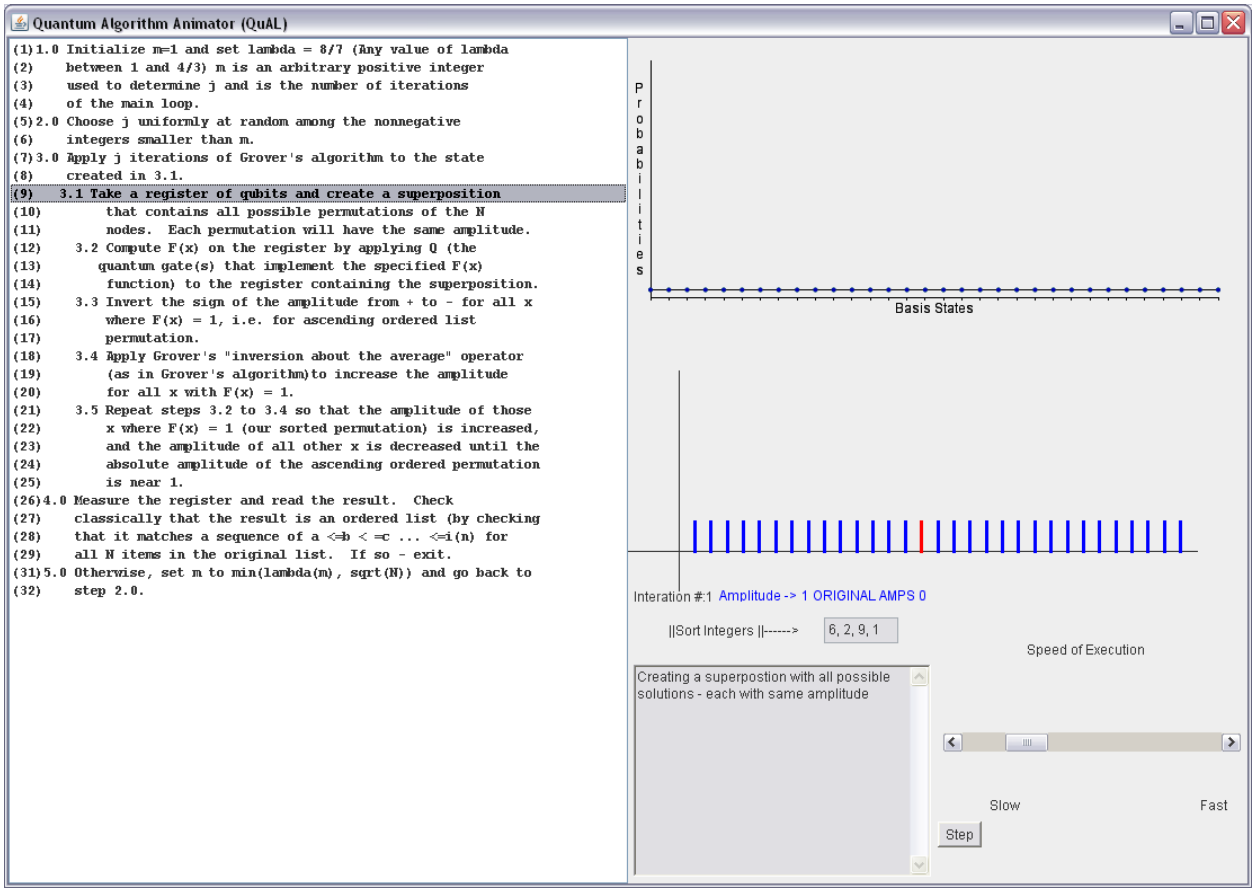
- Input: An unsorted list of N items in arbitrary order;
- Output: A sorted list of N items in a specific sequence;
- Method: A derivative of Grover's algorithm for unstructured database search;

Grover's algorithm has been proven to provide an efficient method for searching unstructured databases by simultaneously examining multiple items in a database in order to answer a single item query. Wallace's algorithm utilizes these foundation quantum principles to search a list of N! permutations, or possible orderings of the N items, where each instance is a single permutation or possible ordering of the N items. The 'marked' permutation satisfies the desired sort sequence.

In this sample run the integers 1,2,6, & 9 are submitted as an unsorted list – [6,2,9,1]. The aim behind this sample run is to use Grover's unstructured search algorithm to carry out an ascending sort of any unordered list by amplifying the desired state (or 'marked' permutation) which represents the ascending sorted list. The function used *f(x)* will return the value 1 (or true) when the values are sorted in the desired output sequence and it will return 0 (or false) otherwise.

In quantum computing, all search space operations are contained within a blackbox therefore we measure the probability of obtaining the correct response and repeat the algorithm typically for Grover's solution sqrt(2^n) times. This has provided the most efficient response. Running the algorithm too many times will lower the chances just as running it too few times will provide a lower probability of finding the correct 'marked' response.

Starting a sample run:



The basis states are calculated with the following initial probabilities:

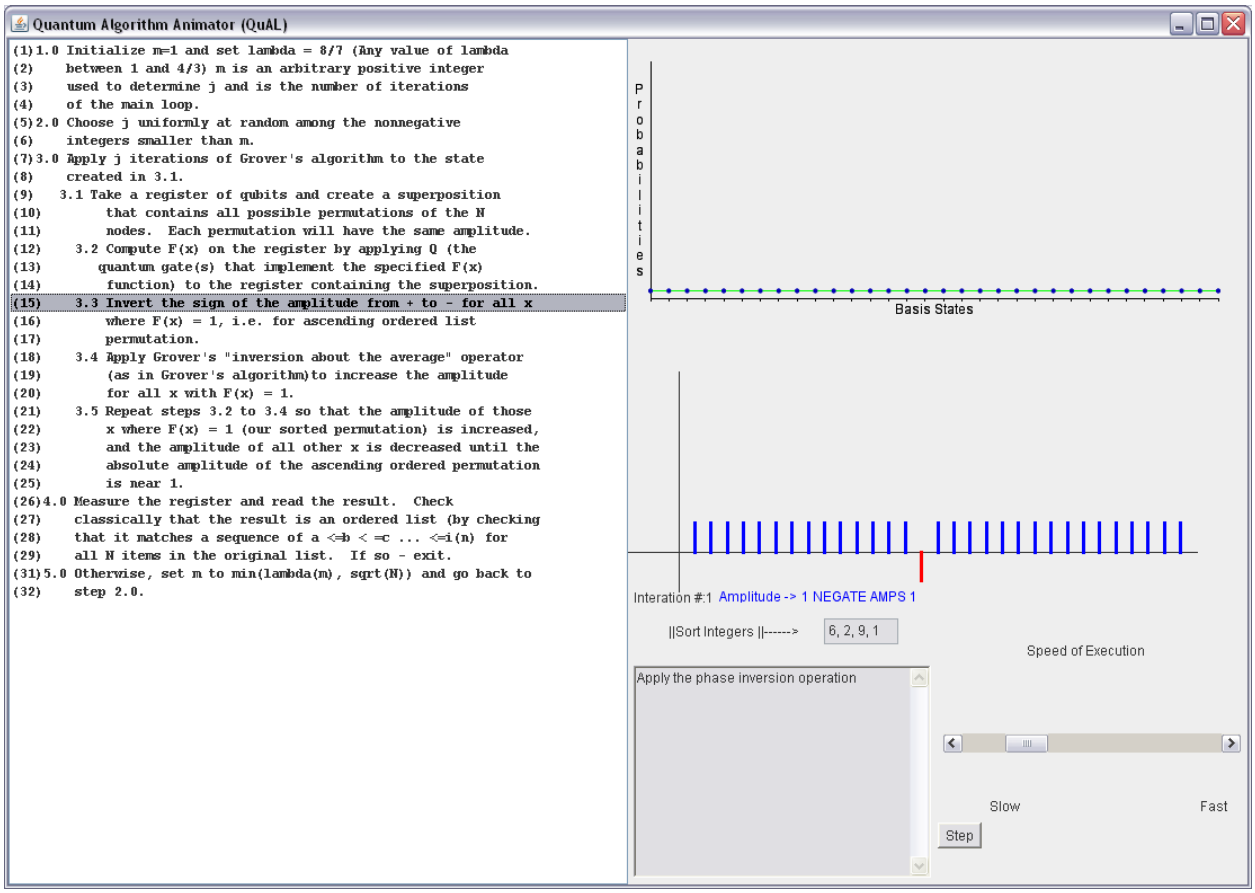
: Initial Probs:
: SPECTRUM q: <0,1,2,3,4>
0.03125 |0>, 0.03125 |1>, 0.03125 |2>, 0.03125 |3>, 0.03125 |4>, 0.03125 |5>, 0.03125 |6>, 0.03125 |7>, 0.03125 |8>, 0.03125 |9>, 0.03125 |10>, 0.03125 |11>, 0.03125 |12>, 0.03125 |13>, 0.03125 |14>, 0.03125 |15>, 0.03125 |16>, 0.03125 |17>, 0.03125 |18>, 0.03125 |19>, 0.03125 |20>, 0.03125 |21>, 0.03125 |22>, 0.03125 |23>, 0.03125 |24>, 0.03125 |25>, 0.03125 |26>, 0.03125 |27>, 0.03125 |28>, 0.03125 |29>, 0.03125 |30>, 0.03125 |31>

Five qubits with three interations are used as a superpostion of all states provides the following initial amplitudes:

STATE: 6 / 32 qubits allocated, 26 / 32 qubits free
:Initial Amplitudes:
0.17678 |0> + 0.17678 |1> + 0.17678 |2> + 0.17678 |3> + 0.17678 |4> + 0.17678 |5> + 0.17678 |6> + 0.17678 |7> + 0.17678 |8> + 0.17678 |9> + 0.17678 |10> + 0.17678 |11> + 0.17678 |12> + 0.17678 |13> + 0.17678 |14> + 0.17678 |15> + 0.17678 |16> + 0.17678 |17> + 0.17678 |18> + 0.17678 |19> + 0.17678 |20> + 0.17678 |21> + 0.17678 |22> + 0.17678 |23> + 0.17678 |24> + 0.17678 |25> + 0.17678 |26> + 0.17678 |27> + 0.17678 |28> + 0.17678 |29> + 0.17678 |30> + 0.17678 |31>

The ‘marked’ amplitude is designated by the red bar while all the other amplitudes are blue. This provides a graphical representation showing how the amplitude is treated during the run of the quantum algorithm.

The main idea behind Grover’s algorithm is to magnify the amplitude of the ‘marked’ number until the probability that it will be our solution is greater then 50%. To do this we use phase inversion and negation to find the ‘marked’ solution and then Grover’s inversion about the mean operation. This operation is repeated until the desired amplitude is reached.



This screen provides a visual of the first interation of probabiliies and amplitudes after the Negate phase with the following amplitude values:

: AfterNegate:Amps:

0.17678 |0> + 0.17678 |1> + 0.17678 |2> + 0.17678 |3> + 0.17678 |4> + 0.17678 |5> + 0.17678 |6> + 0.17678 |7> + 0.17678 |8> + 0.17678 |9> + 0.17678 |10> + 0.17678 |11> + 0.17678 |12> + 0.17678 |13> - 0.17678 |14> + 0.17678 |15> + 0.17678 |16> + 0.17678 |17> + 0.17678 |18> + 0.17678 |19> + 0.17678 |20> + 0.17678 |21> + 0.17678 |22> + 0.17678 |23> + 0.17678 |24> + 0.17678 |25> + 0.17678 |26> + 0.17678 |27> + 0.17678 |28> + 0.17678 |29> + 0.17678 |30> + 0.17678 |31>

With the following probabilities:

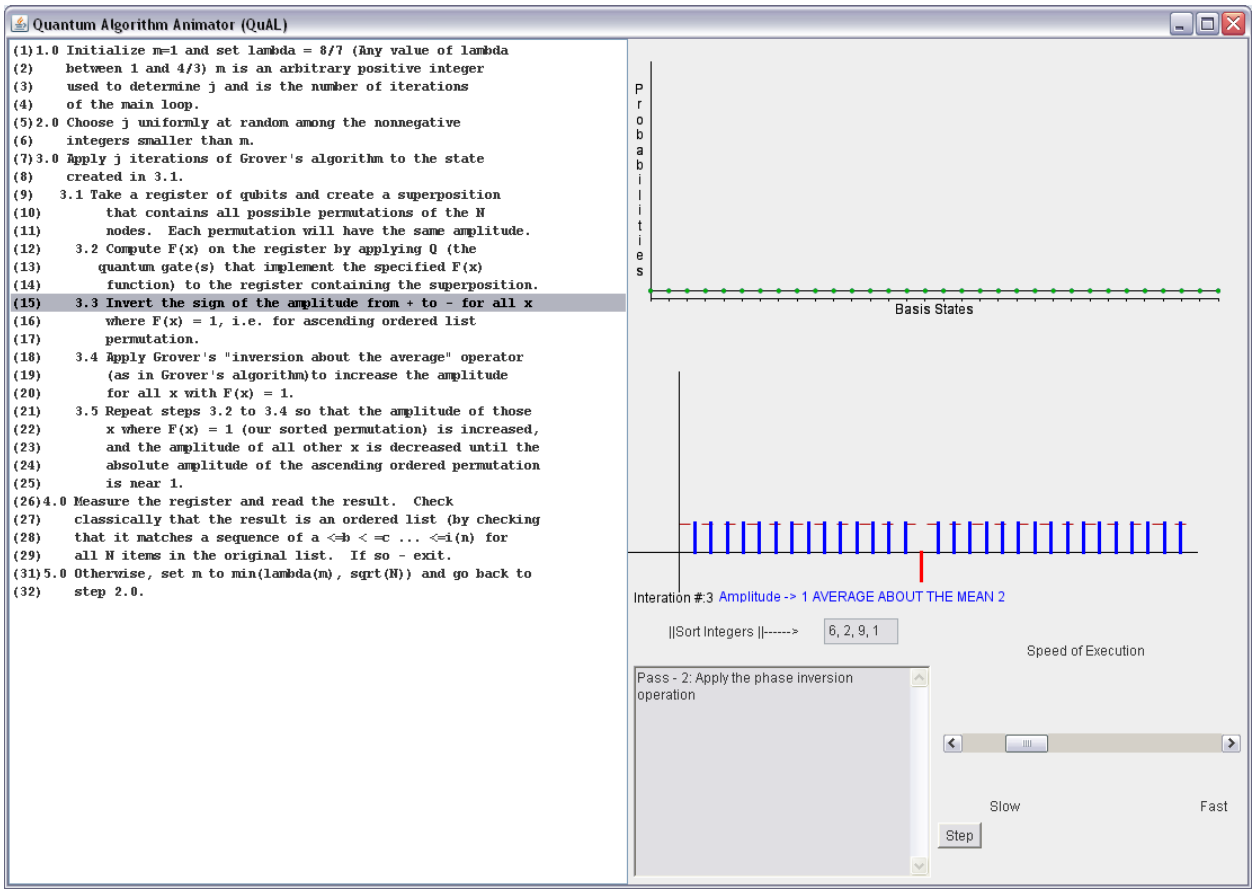
: AfterNegate:Probs:

0.03125 |0>, 0.03125 |1>, 0.03125 |2>, 0.03125 |3>, 0.03125 |4>, 0.03125 |5>, 0.03125 |6>, 0.03125 |7>, 0.03125 |8>, 0.03125 |9>, 0.03125 |10>, 0.03125 |11>, 0.03125 |12>, 0.03125 |13>, 0.03125 |14>, 0.03125 |15>, 0.03125 |16>, 0.03125 |17>, 0.03125 |18>, 0.03125 |19>, 0.03125 |20>, 0.03125 |21>, 0.03125 |22>, 0.03125 |23>,

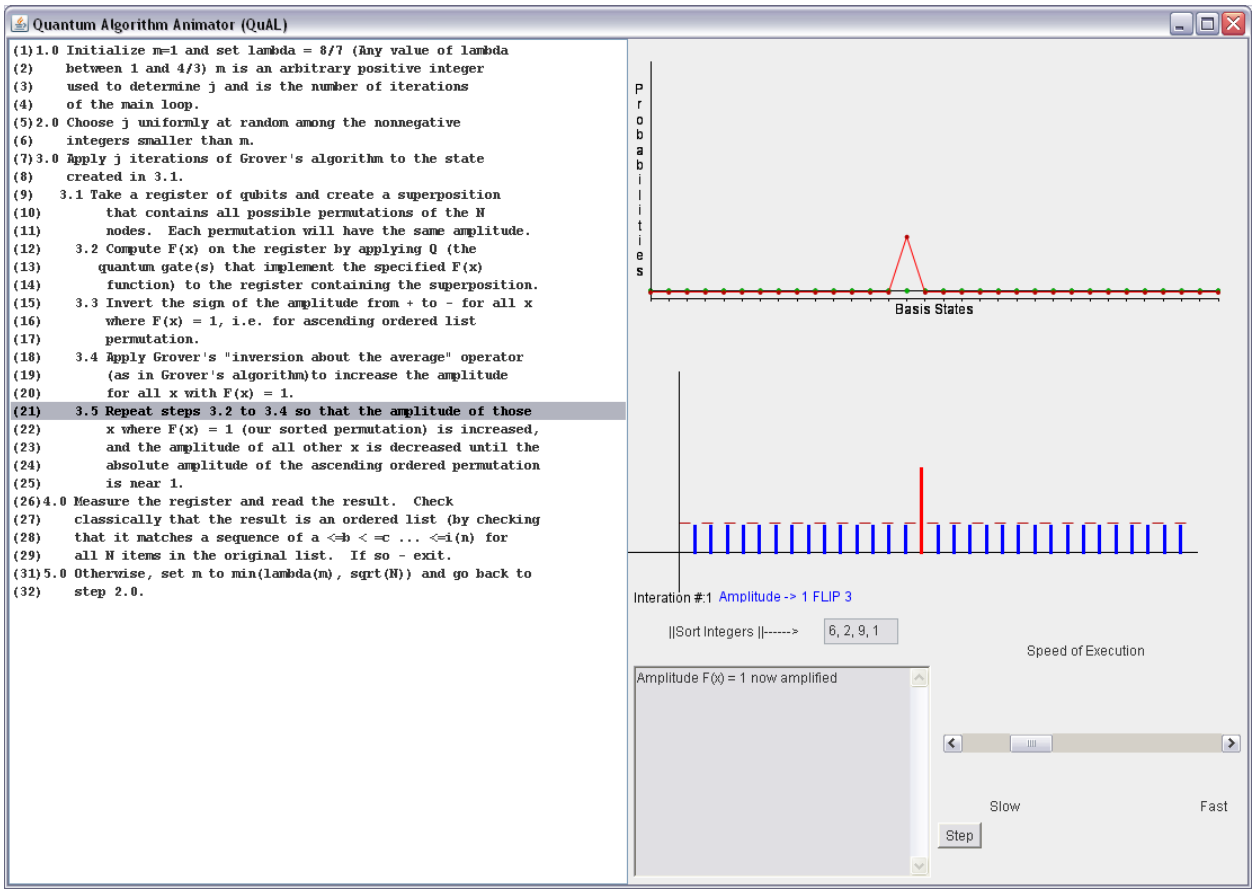
0.03125 |24>, 0.03125 |25>, 0.03125 |26>, 0.03125 |27>, 0.03125 |28>, 0.03125 |29>, 0.03125 |30>, 0.03125 |31>

The amplitudes squared provide the probabilities of measuring those numbers.

The next step is inversion about the mean operations. Here is a screen shot of how each will respond to this calculation:



The dashed red line in the Amplitude graph denotes the average.

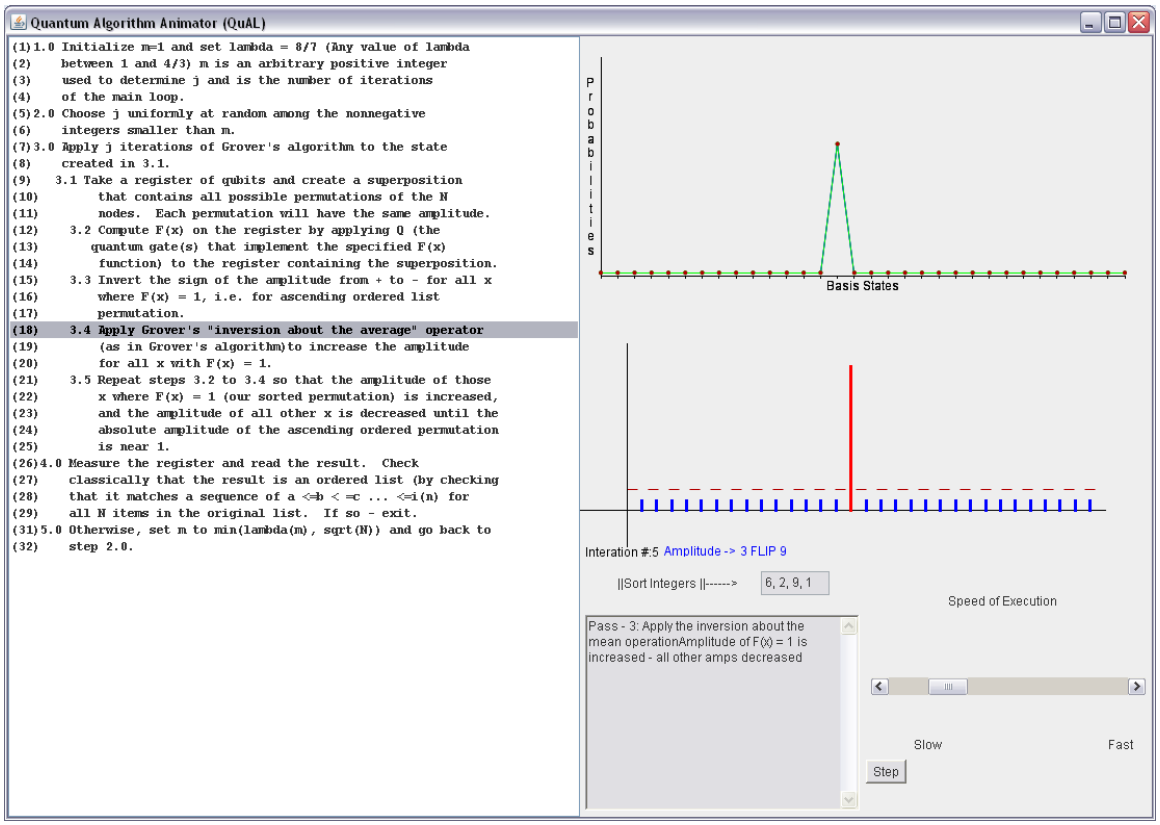


In this screen shot, the probabilities are spiking for the ‘marked’ solution and the amplitude is amplified for the ‘marked’ solution after the average about the mean operation.

: Diffusion:Probs:

0.023926 |0>, 0.023926 |1>, 0.023926 |2>, 0.023926 |3>, 0.023926 |4>, 0.023926 |5>, 0.023926 |6>, 0.023926 |7>, 0.023926 |8>, 0.023926 |9>, 0.023926 |10>, 0.023926 |11>, 0.023926 |12>, 0.023926 |13>, 0.2583 |14>, 0.023926 |15>, 0.023926 |16>, 0.023926 |17>, 0.023926 |18>, 0.023926 |19>, 0.023926 |20>, 0.023926 |21>, 0.023926 |22>, 0.023926 |23>, 0.023926 |24>, 0.023926 |25>, 0.023926 |26>, 0.023926 |27>, 0.023926 |28>, 0.023926 |29>, 0.023926 |30>, 0.023926 |31>

These procedures are repeated three iterations for this algorithm until the final screen shot provides:



With the following results:

: Diffusion:Amps:

-0.05766 |0> - 0.05766 |1> - 0.05766 |2> - 0.05766 |3> - 0.05766 |4> - 0.05766 |5> - 0.05766 |6> - 0.05766 |7> - 0.05766 |8> - 0.05766 |9> - 0.05766 |10> - 0.05766 |11> - 0.05766 |12> - 0.05766 |13> - 0.94707 |14> - 0.05766 |15> - 0.05766 |16> - 0.05766 |17> - 0.05766 |18> - 0.05766 |19> - 0.05766 |20> - 0.05766 |21> - 0.05766 |22> - 0.05766 |23> - 0.05766 |24> - 0.05766 |25> - 0.05766 |26> - 0.05766 |27> - 0.05766 |28> - 0.05766 |29> - 0.05766 |30> - 0.05766 |31>

: Diffusion:Probs:

0.0033246 |0>, 0.0033246 |1>, 0.0033246 |2>, 0.0033246 |3>, 0.0033246 |4>,
0.0033246 |5>, 0.0033246 |6>, 0.0033246 |7>, 0.0033246 |8>, 0.0033246 |9>,
0.0033246 |10>, 0.0033246 |11>, 0.0033246 |12>, 0.0033246 |13>, 0.89694 |14>,
0.0033246 |15>, 0.0033246 |16>, 0.0033246 |17>, 0.0033246 |18>, 0.0033246 |19>,
0.0033246 |20>, 0.0033246 |21>, 0.0033246 |22>, 0.0033246 |23>, 0.0033246 |24>,
0.0033246 |25>, 0.0033246 |26>, 0.0033246 |27>, 0.0033246 |28>, 0.0033246 |29>,
0.0033246 |30>, 0.0033246 |31>

Our probability is now greater than 50% so we can measure our system :

: measured: 14
[0/32] 1 |0>
Checking our permutations classically – the sorted sequence is:
(0) 6 2 1 9
(1) 6 2 9 1
(2) 6 1 2 9
(3) 6 1 9 2
(4) 6 9 2 1
(5) 6 9 1 2
(6) 2 6 1 9
(7) 2 6 9 1
(8) 2 1 6 9
(9) 2 1 9 6
(10) 2 9 6 1
(11) 2 9 1 6
(12) 1 6 2 9
(13) 1 6 9 2
(14) 1 2 6 9
(15) 1 2 9 6
(16) 1 9 6 2
(17) 1 9 2 6
(18) 9 6 2 1
(19) 9 6 1 2
(20) 9 2 6 1
(21) 9 2 1 6
(22) 9 1 6 2
(23) 9 1 2 6
(24)null
(25)null
(26)null
(27)null
(28)null
(29)null
(30)null
(31)null

Appendix J
Wallace and Narayanan's Quantum Sorting Algorithm

1.0 Initialize m=1 and set lambda = 8/7 (Any value of lambda between 1 and 4/3) m is an arbitrary positive integer used to determine j and is the number of iterations of the main loop.

2.0 Choose j uniformly at random among the nonnegative integers smaller than m.

3.0 Apply j iterations of Grover's algorithm to the state created in 3.1.

3.1 Take a register of qubits and create a superposition that contains all possible permutations of the N nodes. Each permutation will have the same amplitude.

3.2 Compute F(x) on the register by applying Q (the quantum gate(s) that implement the specified F(x) function) to the register containing the superposition.

3.3 Invert the sign of the amplitude from + to - for all x where F(x) = 1, i.e. for ascending ordered list permutation.

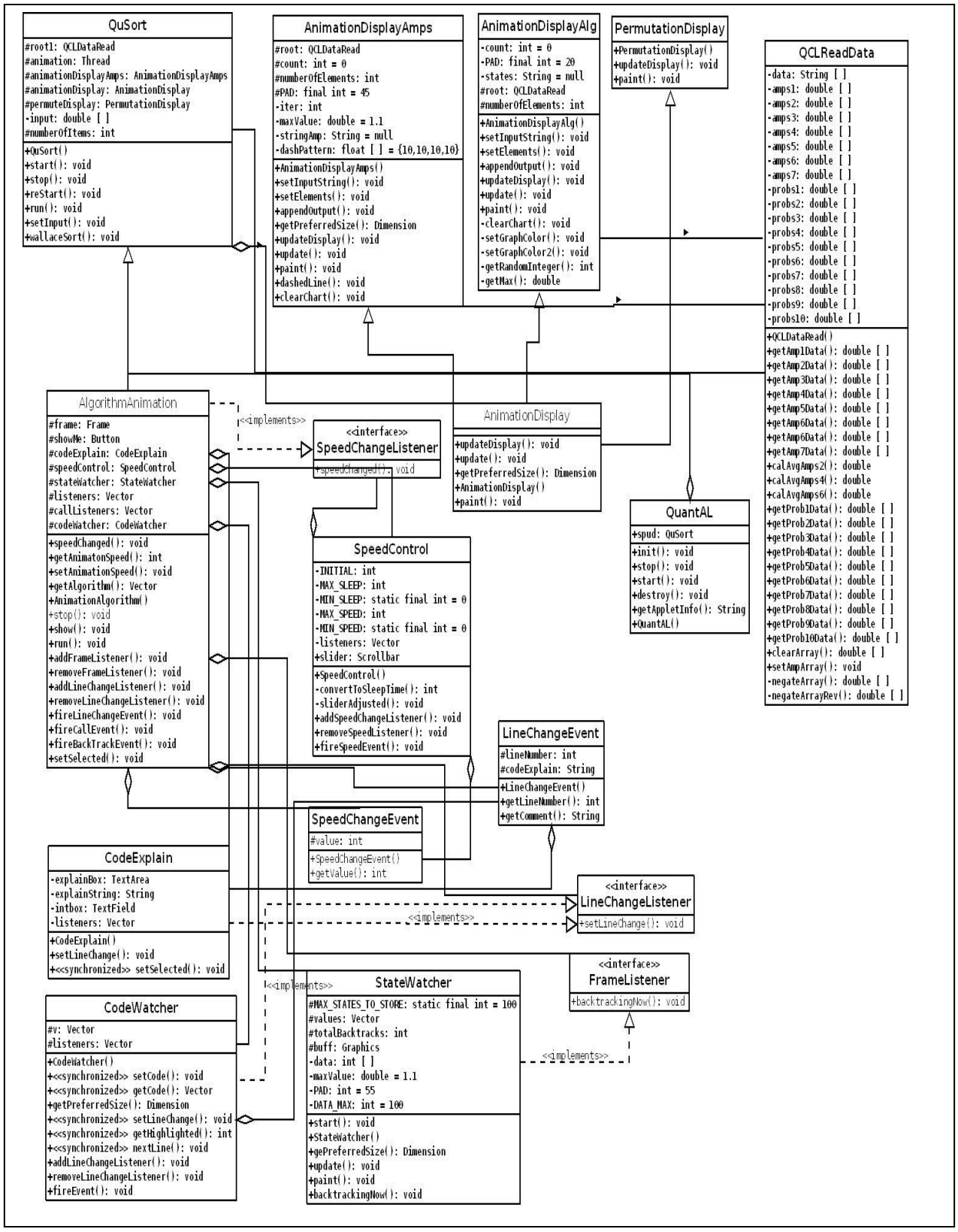
3.4 Apply Grover's "inversion about the average" operator (as in Grover's algorithm)to increase the amplitude for all x with F(x) = 1.

3.5 Repeat steps 3.2 to 3.4 so that the amplitude of those x where F(x) = 1 (our sorted permutation) is increased, and the amplitude of all other x is decreased until the absolute amplitude of the ascending ordered permutation is near 1.

4.0 Measure the register and read the result. Check classically that the result is an ordered list (by checking that it matches a sequence of a <=b < =c ... <=i(n) for all N items in the original list. If so - exit.

5.0 Otherwise, set m to min(lambda(m), sqrt(N)) and go back to step 2.0.

Appendix K
UML Class Diagram of QuAL



Reference List

Aharonov, D. (1998, December 15). *Cornell University Library*. Retrieved November 1, 2010, from arXiv.org: http://arxiv.org/abs/quant-ph/9812037

Aharonov, D., & Ta-Shma, A. (2003). Adiabatic quantum state generation and statistical zero knowledge. *Proceedings of the thrity-fifth annual ACM symposium on Theory of computing* (pp. 20-29). San Diego: ACM.

Aharonov, y. D., & Zagury, N. (1993). Quantum random walks. *Phys. Rev.*.

Ambainis, A. (2005, October 19). *Cornell University Library*. Retrieved January 28, 2010, from arXiv.org: http://arxiv.org/PS_cache/quant-ph/pdf/0311/0311001v8.pdf

Ambainis, A. (2000). Quantum lower bounds by quantum arguments. *32nd ACM Symposium on Theory of Computing* , 636-643.

Ambainis, A., Kempe, J., & Rivosh, A. (2005). Coins make faster quantum walks. *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* , 1099-1108.

Ayellet, T. (1994). *Animation and visualization of geometric algorithms.* Princeton University. Available from Dissertations and Theses database.

Bacon, D., & Leung, D. (2007). Toward a World with Quantum Computers. *Communications of the ACM , 50* (9), 55 - 59.

Badre, A., Beranek, M., Morris, J., & Stasko, J. (1991). Assessing program visualization systems as instructional aids. *Georgia Institute of Techology GVU-91-23* .

Baecker, R. (Director). (1981). *Sorting out sorting* [Motion Picture].

Bell, J. (1964). On the Einstein-Podolsky-Rosen Paradox. *Physics*, 195-200.

Bennink, R. (2008). Quantum information: opportunities and challenges or "what's this quantum stuff and what does it have to do with cyber security". *Proceedins of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead* (p. 27). Oak Ridge: ACM.

Brassard, G. (1997). Searching a quantum phone book. *Science* , 627 - 628.

Brassard, G., & Hoyer, P. (1997). An Exact Quantum Polynomial-Time Algorithm for Simon's Problem. *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems (ISTCS'97)* .

Brassard, G., Hoyer, P., & Tapp, A. (1997). Quantum algorithm for the collision problem. *ACM SIGACT News* , 14-19.

Brassard, G., Hoyer, P., Mosca, M., & Tapp, A. (2000). Quantum amplitude amplification and estimation. *quant-ph/0005055* .

Brown, M. (1988). Exploring algorithms with Balsa-II. *Computer* , 14-36.

Brown, M., & Hershberger, J. (1991). Color and sound in algorithm animation. *Proceedings of the 1991 IEEE Workshop on Visual Languages* (pp. 10-17). Los Alamitos: IEEE CS Press.

Brown, M., & Hershberger, J. (1991). Zeus: a system for algorith animation. *Proceedings IEEE Workshop in Visual Languages* (pp. 4-9). Los Alamitos: IEEE CS Press.

Brown, M., & Sedgewick, R. (1974). A system for algorithm animation. *Proceedings of ACM SIGGRAPH '84* (pp. 177-186). New York: ACM.

Bryne, M., Catrambone, R., & Stasko, J. (1999). Evaluating animation as student aids in learning computer algorithms. *Computers and Education* , 253-278.

Buhrman, H., & de Wolf, R. (2002). Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science* , 21-43 .

Buhrman, H., & Spalek, R. (2005). Quantum verification of matrix products. *In Proceedings of the 17th annual ACM-SIAM symposium on Discrete Algorithms*, p.880.

Buhrman, H., Durr, C., Heiligman, M., Hoyer, P., Magniez, F., Santha, M., et al. (2001). Quantum algorithms for element distinctness. *Proceedings of the 16th Annual Conference on Computational Complexity (CCC'01)* .

Cattaneo, G., Ferraro, U., Italiano, R., & Scarano, V. (2002). Concurrent algorithms and data types animation over the internet. *Journal of Visual Languages & Computing* , 391-419.

Chen, T., & Sobh, T. (2001). A tool for data structure visualization and user-defined algorithm animation. *Frontiers in Education Conference, 31st Annual* (pp. 2-7). IEEE.

Ciesielski, V., & McDonald, P. (2001). Using animation of state space algorithms to overcome student learning difficulties. *Proceedings of the 6th annual conference on Innovation and technology in computer science education* (pp. 97-100). Canterbury: ACM.

Crescenzi, P., Demetrescu, C., Finocchi, I., & Petreschi, R. (1997). LEONARDO: a softward visualization system. *Proceedings 1st Workshop on Algorithm Engineering*, (pp. 146-155). Venice.

Creswell, J. (2002). *Educational research: Planning, conducting, and evaluating quantitative and qualitative approaches to research.* Upper Saddle River, NJ: Merrill / Pearson Education.

Creswell, J. (2003). *Research design: Qualitative, quantitative, and mixed methods approaches* (2nd Edition ed.). Thousand Oaks, CA: Sage Publications.

Demetrescu, C., Finoci, I., & Stasko, J. (2001). Specifying algorithm visualizations: Interesting events or state mapping? In S. Diehl, *Software Visualization: International Seminar* (pp.16-30). Dagstuhl,Germany:Springer.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London, Series A,* (pp. 97-117). London.

Deutsch, D., & Jozsa, R. (1992). Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London, Series A,* (pp. 553- 558). London.

Dickau, R.M. (1996). Permutation Diagrams. Retrieved November 1, 2010 from http://mathforum.org/advanced/robertd/permutation.html.

Diehl, S. (2007). *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software.* New York: Springer.

Dobkin, D. (1992). Computational geometry and computer graphics. *Journal Proceedings of the IEEE , 80* (9), 1400-1411.

Durr, C., & Hoyer, P. (1999, January 7). *University of Cornell Library*. Retrieved November 1, 2010, from axtiv.org: http://arxiv.org/abs/quant-ph/9607014.

Einstein, A., Podolsky, B., Rosen, N.(1935). Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Physical Review,* pp. 777.

Emberson, P. (2002). *An introduction to quantum algorithm design.* Heslington: Department of Computer Science, University of York.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading: Addison Wesley.

Gloor, P. (1997). *Elements of hypermedia design.* Boston: Birkhäuser.

Grillmeyer, O. (2001). *Designing Effective Animations for Computer Science Instruction.* Berkeley: University of California.

Grisson, S., McNally, M., & Naps, T. (2003). Algorithm visualization in computer science education: comparing levels of student engagement. *SoftVis '03: Proceedings of the 2003 ACM symposium of Software Visualization* (pp. 87-94). San Diego: ACM.

Grover, L. (1996). A faster quantum mechanical algorithm for database search. *STOC '96: Proceedings of the 28th Annual ACM Symposium of Theory of Computing* (pp. 212-219). New York: ACM.

Gurka, J. (1997). *Pedagogic aspects of algorithm animation.* University of Colorado. Boulder: Available from Dissertations and Theses database.

Hansen, S., Schrimpsher, D., & Narayanan, N. (1998). *Empirical studies of animation-embedded hypermedia algorithm visualizations.* Auburn: Auburn University.

Hardy,Y., & Steeb, W.(2001). *Classical and Quantum Computing.* Basel, Switzerland: Birkhauser Verlag.

Heer, J., Card, S., & Landay, J. (2005). prefuse: a toolkit for interactive information visualization. *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 421-430). Portland: ACM.

Hey, T., Walters, P.(1987). *The Quantum Universe.* Cambridge: Cambridge University Press.

Hogg,T. (1997, January 13). *Cornell University Library*. Retrieved November 1, 2010, from http://arxiv.org/abs/quant-ph/9701013v1

Hopgood, F. (1974). Computer animation used as a tool in teaching computer science. *Proceedings of the IFIP Congress* (pp. 889-892). Stockholm: IFIP.

Hoyer, P., Neerbek, J., & Shi, Y. (2002). Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica* , 429-448.

Hundhausen, C., Douglas, S., & Stasko, J. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Language Computation , 13* (3), 259–290.

Jackson, D., & Fovargue, A. (1997, May). The use of animation to explain genetic algorithms. *SIGCE '97* .

Jones, D., & Newman, A. (1996). RCOS.java: an animated operating system for computer science education. *Annual Joint Conference Integrating Technology into Computer Science Education*, (p. 233). Bracelona.

Jorrand, P. (2007). Quantum information processing and communication: the computer science perspective. *Proceedings of the 45th annual southeast regional conference* (p. 509). Winston-Salem: ACM.

Kasivajhula, S. (2006). Quantum Computing: A Survey. *ACM SE 2006* , 248-253.

Kehoe, C., & Stasko, J. (1996). *Using animations to learn about algorithms: an ethnographic case study.* Atlanta: Georgia Institute of Technology.

Kehoe, C., Stasko, J., & Taylor, A. (2001). Rethinking the evaluation of algorithm animations as learning aids: an observational study. *International Journal of Human-Computer Studies* , 265-284.

Kerran, A., & Stasko, J. (2002). Algorithm Animation: Introduction. In S. Diehl, *Software Visualization* (pp. 1-15). Springer-Verlag.

Kitchenham, B., Pickard, L., & Pfleeger, S. (1995). Case studies for method and tool evaluation. *IEEE Software* , 52-62.

Klauck, H. (2003). Quantum time-space tradeoffs for sorting. *STOC'03* .

Koldehofe, B., Papatriantafilou, M., & Tsigas, P. (2006). LYDIAN: An extensible educational animation environment for distributed algorithms. *ACM Journal on Educational Resource in Computing , 6* (2), 1-21.

Laszlo, M. (2002). *Object-oriented programming featuring graphical appliations in Java.* Boston: Addison Wesley.

Latombe, J. (1991). *Robot Motion Planning.* Kluwer Academic Publishers.

Leung, K. (2005). *Animation of linux processor scheduling algorithm, master's degree thesis.* Sacramento: Department of Computer Science, California State University.

Levitin, A. (1999). Do we teach the right algorithm design techniques? *The proceedings of the thirtieth SIGCSE technical symposium on Computer science education* (pp. 179-183). New Orleans: ACM.

Magniez, F., Santha, M., & Szegedy, M. (2005). Quantum algorithms for the triangle problem. *In Proceedings of the 16th Annual ACM SIAM symposium on Discrete Algorithms* , 1109.

Mermin, D. (2006). *Breaking RSA encryption with a quantum computer: Shor's factoring algorithm.* Ithaca: Cornell University.

Nielsen,M., & Chuang,I.(2000). *Quantum Computation and Quantum Information.* New York:Cambridge University Press.

Omer, B. (1998). *A Procedural Formalism for Quantum Computing.* Vienna:Department of Theoretical Physics, Technical University of Vienna.

Plaisant, C., & North, C. (2007). Reflections on Human-Computer Interaction. *International Journal of Human-Computer Interaction , 23* (8), 195 - 204.

Plaisant, C., & Vinit, J. (1994). Dynamaps: dynamic queries on a health statistics atlas. *Conference companion on Human factors in computing systems* (pp. 439-440). Boston: ACM.

Price, B., Baecker, R., & Small, I. (1998). An introduction to software visualization. In J. Stasko, J. Domingue, M. Brown, & B. Price, *Software Visualization* (pp. 3-27). Cambridge: MIT Press.

Rasala, R., Proulx, V., & Fell, H. (1994). From animation to analysis in introductory computer science. *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education* (pp. 61-65). Phoenix: ACM.

Rieffel,E.,&Polak,W.(2000). An Introduction to Quantum Computing for Non-Physicists. *ACM Computing Surveys*, pp. 300-325.

Robling, G., Schuer, M., & Freisleben, B. (2000). The ANIMAL algorithm animation tool. *Annual Joint Conference Integrating Technology into Computer Science* (pp. 37-40). Helsinki: ACM.

Robson, C. (2002). *Real World Research* (2nd Edition ed.). Blackwell.

Rose, A., Shneiderman, B., & Plaisant, C. (1995). An applied ethnographic method for redesigning user interfaces. *DIS '95: Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques* (pp. 115 - 122). Ann Arbor: ACM.

Runeson, P., & Host, M. (2008, 12 19). Retrieved November 1, 2010, from Springerlink.com: http://www.springerlink.com/content/t22r8l65q7h31636/

Saeedi, M., Zamani, M., & Sedighi, M. (2008). Moving forward: a non-search based synthesis method towards efficient CNOT-based quantum circuit synthese algorithms. *Proceedings of the 2008 Asia and South Pacific Design Automation Conference* (pp. 83-88). Seoul: IEEE Computer Society Press.

Sangwan, R. (1997). *Algorithm animation using self-visualizing C.* Temple University. Available in the dissertation and thesis database.

Schwartz, J., & Yap, C. (1987). *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics.* Hillsdale, N.J.: Lawrence Erlbaum Associates.

Shi, Y. (2001, September 22). *Cornell University Library*. Retrieved November 1, 2010, from arXiv: http://arxiv.org/abs/quant-ph/0009091

Shi, Y. (2002). Entropy lower bounds of quantum decision tree complexity. *Information Processing Letters* , 23-27.

Shneiderman, B. (1983). Direct manipulation: a step beyond programming languages. *IEEE Computer* , 57-69.

Shneiderman, B. (1973). Polynomial Search. *Software: Practice and Experience* , 5-8.

Shneiderman, B., & Plaisant, C. (1998). Information visualization advanced interface and web design. *CHI 98 conference summary on Human factors in computing systems* (pp. 18-23). ACM.

Shneiderman, B., Plaisant, C., Cohen, M., & Jacobs, S. (2009). *Designing the User Inteface: Strategies for Effective Human-Computer Interaction* (5 ed.). Addison-Wesley Publishing Company.

Shor, P. (1994). Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science* (pp. 124-134). IEEE Computer, Society Press.

Shor, P. (1997). Quantum Algorithms. (P. Arneson, Ed.) *Journal of ACM , 1* (2), 16-24.

Shor, P. (2003). Why haven't more quantum algorithms been found? *Journal of the ACM*, 87-90.

Simon, D. (1997). On the power of quantum computation. *SIAM Journal on Computing* , 1474-1483.

Smith, S.,& Mosier,J. (1986). Guidelines for Designing User Interface Software. Available from National Technical Information Service. Springfield,VA:MITRE Corporation.

Stasko, J. (1995). *POLKA animation designer's package.* Atlanta: Georgia Institute of Technology.

Stasko, J. (1990). TANGO: a framework and system for algorithm animation. *Computer*, 27-39.

Stasko, J. (1997). Using student-built algorithm animations as learning aids. *Proceedings of ACM SIGSE Technical Symposium on Computer Science Education* (pp. 25-29). New York: ACM Press.

Stasko, J., Badre, A., & Lewis, C. (1993). Do algorithm animations assist learning? An empirical study and analysis. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 61-66). Los Angeles: ACM.

Stasko, J., Dominque, J., Brown, M. H., & Price, B. A. (1997). *Software Visualization.* Massachusetts Institute of Technology.

(2002). *Theory of quantum computing and communication.* Elmsford: National Science Foundation.

Toussaint, G. (1986). New results in computational geometry relevant to pattern recognition in practice. *Pattern Recognition in Practice II* , 135-146.

Tory, M., & Moller, T. (2004). Human Factors in Visualization Research. *IEEE Transactions on Visualization and Computer Graphics.*10, pp. 72-84. IEEE.

Tudoreanu, M., Wu, R., Hamilton-Taylor, A., & Kraemer, E. (2002). Empirical evidence that algorithm animation promotes understanding of distributed algorithms. *Proceedings of the IEEE 2002 Symposium on Human Centric Computing Languages & Environment* (p. 245). Arlington: IEEE.

Urquiza-Fuentes, J., & Velazquez-Iturbide, J. (2009). A survey of successful evaluations of program visualizations and algorithm animation systems. *ACM Transactions on Computer Education TOCE , 9* (2).

Vrachnos, E., & Jimoyiannis, A. (2008). DAVE: A Dynamic Algorithm Visualization Environment for Novice Learners. *2008 Eighth IEEE International Conference on Advanced Learning Technologies* (pp. 319-323). Cantabria: IEEE Computer Society.

Wallace, J., & Narayanan, A. (2001). Quantum sorting and route finding. *Computing Anticipatory Systems: CASYS 2001.* American Institue of Physics.

Walsham, G. (1993). *Interpreting information systems in organizations.* London: John Wiley and Sons.

Ware, C. (2000). *Information visualization: perception for design.* San Franciso: Morgan Kaufmann Publishers Inc.

Wiggins, M. (1998). *A comparative study of two algorithm animation tools as aids in algorithm understanding.* University of Southern Mississippi, UMI No. 9840847. Available from Dissertations and Theses database.

Yanofsky, N., & Mannucci, M. (2008). *Quantum Computing for Computer Scientists.* New York: Cambridge.

Yao, A. (1994). Near optimal time-space tradeoffs for element distinctness. *SIAM Journal on Computing* , 996-975.

Yin, R. (2003). *Case study research* (3rd Edition). London: Sage.

Zhou, X., Li, W., Xian, H., Lai, T., & Liang, H. (2008). Towards context modeling for algorithm animation. *32nd Annual IEEE International Computer Software and Applications Conference* (pp. 279-286). IEEE.