COMPUTATIONAL THINKING AND WOMEN IN COMPUTER SCIENCE

by

CHRISTIE LEE LILI PROTTSMAN

A THESIS

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

June 2011

THESIS APPROVAL PAGE

Student: Christie Lee Lili Prottsman

Title: Computational Thinking and Women in Computer Science

This thesis has been accepted and approved in partial fulfillment of the requirements for the Master of Science degree in the Department of Computer and Information Science by:

Michal Young	Chairperson
Joanna Goode	Member

and

Richard Linton	Vice President for Research and Graduate Studies/Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded June 2011

THESIS ABSTRACT

Christie Lee Lili Prottsman

Master of Science

Department of Computer and Information Science

June 2011

Title: Computational Thinking and Women in Computer Science

Approved: _____
Michal Young

Though the first computer programmers were female, women currently make up only a quarter of the computing industry. This lack of diversity jeopardizes technical innovation, creativity and profitability. As demand for talented computing professionals grows, both academia and industry are seeking ways to reach out to groups of individuals who are underrepresented in computer science, the largest of which is women.

Women are most likely to succeed in computer science when they are introduced to computing concepts as children and are exposed over a long period of time. In this paper I show that computational thinking (the art of abstraction and automation) can be introduced earlier than has been demonstrated before. Building on ideas being developed for the state of California, I have created an entertaining and engaging educational software prototype that makes primary concepts accessible down to the third grade level.

CURRICULUM VITAE

NAME OF AUTHOR: Christie Lee Lili Prottsman

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene

DEGREES AWARDED:

Master of Science, Computer and Information Science, 2011, University of Oregon
Bachelor of Science, Mathematics, 1999, University of Oregon

AREAS OF SPECIAL INTEREST:

Computational Thinking
Computer Theory
Computer Science Education

PROFESSIONAL EXPERIENCE:

Graduate Research Fellow, Department of Computer and Information Science, University of Oregon, 2009 - 2011

Undergraduate Research Assistant, Department of Computer and Information Science, University of Oregon, 2008 – 2009

Program and Design, Artistry& Programming
Portland, Oregon, 2003 - 2008

ACKNOWLEDGMENTS

I would like to acknowledge Michal Young and Joanna Goode for their help and inspiration throughout the process of completing this thesis. Their motivation and support has contributed greatly to the direction of my studies and for that I thank them immensely.

I also would like to acknowledge the Department of Computer and Information Science at the University of Oregon for providing the opportunities which have led to the research and activities surrounding the work done in this paper.

This thesis is dedicated first and foremost to my sons, James and Jackson Davis who have been the catalysts for everything I have achieved in my life. Also, to my loving family and friends who have believed in me and provided help and unconditional support.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Research has shown that women who succeed in computer science are likely to have been introduced to the field early in life (Camp and Gürer 1997). By familiarizing girls with advanced concepts prior to middle school, advocates hope that their interest in technology will continue to grow during grades six through twelve — instead of showing a significant decrease during those years, as is currently the case (Margolis and Fisher 2002). A strong representation of women in computer science is becoming ever more important as the technology field booms. Female team members not only provide diversity of knowledge and experience, but at 51% of the American population (US Census 2009) and only 26% of the computer science work force (Department of Labor 2010), women are a great resource for adding talent to an area that faces a shortage of qualified applicants. Retaining the interest of girls through high school is the first step toward strengthening their presence in undergraduate study and industry.

In order to more easily achieve that goal, there is a need for tools aimed at presenting true computer science — not just computer literacy — to elementary aged children. This paper focuses on the development of one such tool.

1.1. Contributions

This thesis makes two major contributions to the fields of computer science and computer science education. First, it introduces computational thinking as a promising method for attracting women into computer science.

Second, it provides a tool for introducing computational thinking to an elementary school audience. Computational thinking (thinking like a computer scientist) is currently an extremely popular topic. Blending computational thinking into other fields has already begun to transform the world, especially in the areas of biology and economics (Astrachan 2009).

Previously, educational support in the area of computational thinking has been aimed at advanced grade levels, namely college undergraduates and above. Few methods have reached the K-12 age group and most of those are largely hypothetical, giving scenarios for learning opportunities without providing concrete instructional tools (such as Barr, Harrison, and Conery 2011). This thesis introduces not only a self-contained online computational thinking tool for grades three through five, but also provides printed examples of analogous lessons for those without access to computers.

1.2. Organization of Thesis

The remainder of this thesis is organized as follows: Chapter II focuses on the history of participation in computer science, highlighting the widening gap between the number of men and women in the field. Chapter III reviews existing research surrounding the reasons why technological degrees and careers fail to appeal to women. Chapter IV discusses lack of early opportunity as the root problem for women in the industry and introduces computational thinking as a solution to that problem. Chapter V presents Thinking Myself, an online game designed to cultivate confidence and resilience among girls in computer science. To conclude, chapter VI summarizes the ideas presented in this paper and suggests future areas of research.

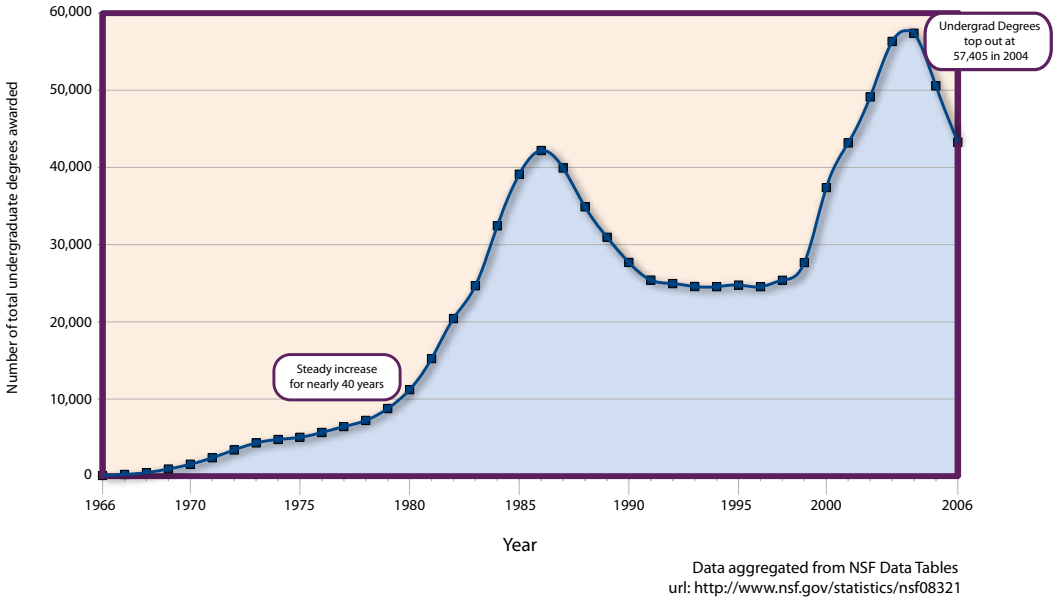
CHAPTER II

A LOOK AT THE HISTORY OF PARTICIPATION IN COMPUTER SCIENCE

Many people think of computer science as a recent phenomenon, beginning within the last generation or so and peaking in the 1990's before the "dot-com" bust. In reality, the word "computer" has been in use since before the early seventeenth century when it referred to a person who calculated computations (Oxford English Dictionary, Second Edition). Computer science was an important skill for many mathematicians who relied on automation and programming as far back as 100 B.C.(New York Times 2002). Even then, computational theory and algorithms were critically important, providing methods to solve difficult problems in the least amount of time with proper preparation.

Computer science evolved from the discipline of math, but as electronic computers became accessible to the public the topic started to be recognized as a valid subject on its own. In the early 1960's, computer science departments began to split themselves out of mathematics departments, with the first actual bachelor's degree in computer science (CS) being awarded at Purdue in 1962 (Rice and Rosen 2004). Enrollment in CS grew steadily for the first decade (National Science Foundation 2008), picking up speed in the late 1970's with the introduction of the 8-bit color, mass-produced Apple II. Between 1978 and 1985, enrollment for undergraduate programs grew almost exponentially, dropping significantly between 1985 and 1991, then eventually stagnating until the late 1990's (see figure 1).

Figure 1. Bachelor degrees awarded by year



In 1996, online activity began to affect interest in computer science as the number of Internet users jumped from 16 million to 36 million (Dreze and Zufryden 1999). The numbers nearly doubled again in 1997 and 1998, leading to a rush in e-commerce activities. This new and profitable model gave birth to a phenomenon called the "dot-com bubble" as investors succumbed to World Wide Web mania, throwing their money into any venture labeled with the ".com" suffix (Goodnight and Green 2010). College enrollments grew with equal fervor, topping out at 57,405 CS bachelor's degrees awarded in 2004 (National Science Foundation 2008). After the dot-com bubble burst in the early twenty-first century, undergraduate enrollments began to decline and have not yet rebounded to previous levels.

2.1. Rich History of Women in CS

The number of women enrolled in undergraduate work in computer science has always trailed the number of men, but that does not mean that

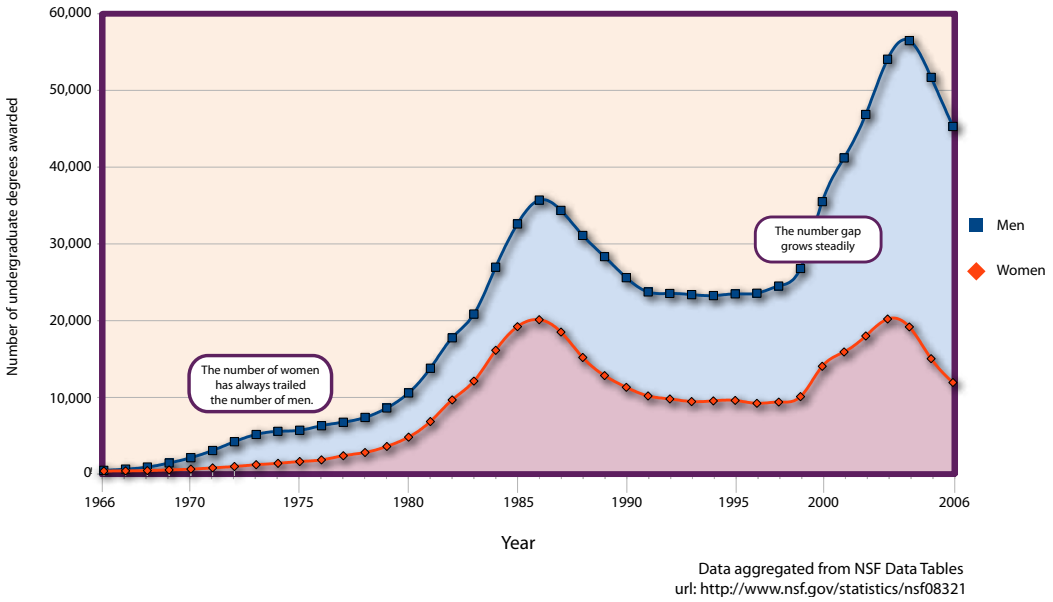
women are less capable of succeeding in the field. As a matter of fact, the first computer programmers were women. As far back as 1843, Ada Lovelace was the first woman to write an algorithm for execution on a machine when she penned some notes for operation of Charles Babbage's hypothetical Analytical Engine (Fuegi and Francis 2003). If speculative instructions intended for an imaginary machine aren't sufficient evidence of a programmer, then the first legitimate computer programmers actually emerged over a century later, and again, they were women.

In 1946, the United States Army publicly introduced the world's first reprogrammable general purpose computer. The Electronic Numerical Integrator And Computer (ENIAC) was capable of making nearly 5000 calculations per second (Farrington 1996, Goldstine 1972). Up to that point, the Army had relied on a group of specially recruited women to calculate trajectory tables for ballistic missiles, a process that took weeks by hand. With the adoption of the ENIAC, the female "computers" — Fran Bilas, Betty Jennings, Ruth Lichterman, Kay McNulty, Betty Snyder, and Marlyn Wescoff — turned their time and intelligence toward programming the thirty ton, 1800 square foot behemoth.

Even with a long history as computational pioneers, women maintain a turbulent relationship with computer science. According to the National Science Foundation (2008), female undergraduates have never made up more than 37% of the CS major (see figure 2). Despite the fact that colleges and universities are running campaigns targeting women, their numbers continue to drop (U.S. Department of Education 2009). The attrition rate among women in CS departments is astounding. Only 40% of women who begin in the CS program will graduate with that major, compared to 68% of men (Klawe and Levenson

1995). Women tend to perceive themselves as outsiders and "imposters" (Schenkel 1984), making them far less likely to hang on through daily struggles as they believe that they should never have started in the first place.

Figure 2. Bachelor's degrees awarded to women vs. men



With women making up nearly half of the overall workforce (U.S. Department of Labor 2009), one might question why it is so important to equalize the gender gap in computer science specifically. Women are responsible for 58% of e-commerce spending (Abraham, Mörn, and Vollman 2010) and account for 55% of social gamers (Information Solutions Group 2010). In many respects, women make up as much or more of the end-user demographic than men do, and yet 79% of those designing the services and 78% of those creating them are male (Department of Labor 2010). Equalizing this imbalance, however, would do far more than just enlighten corporations to the desires of their customers. Diversified teams inspire more teamwork, creativity and productivity than unisex teams (Ashcraft and Blithe 2009). Companies with the highest

percentage of women in upper management experience a return of equity that is 35% higher than those corporations with the lowest percentage of women at executive levels (Joy et al. 2007).

Knowing the success that women have previously achieved as pioneers in computer science makes it that much more baffling to find that so few of them are involved in the field today. The next section will summarize research that gives insight into the problems, preferences and rumors that contribute to fewer women pursuing and succeeding in technological careers.

CHAPTER III

SPECULATIONS AND ATTEMPTS AT SOLUTIONS

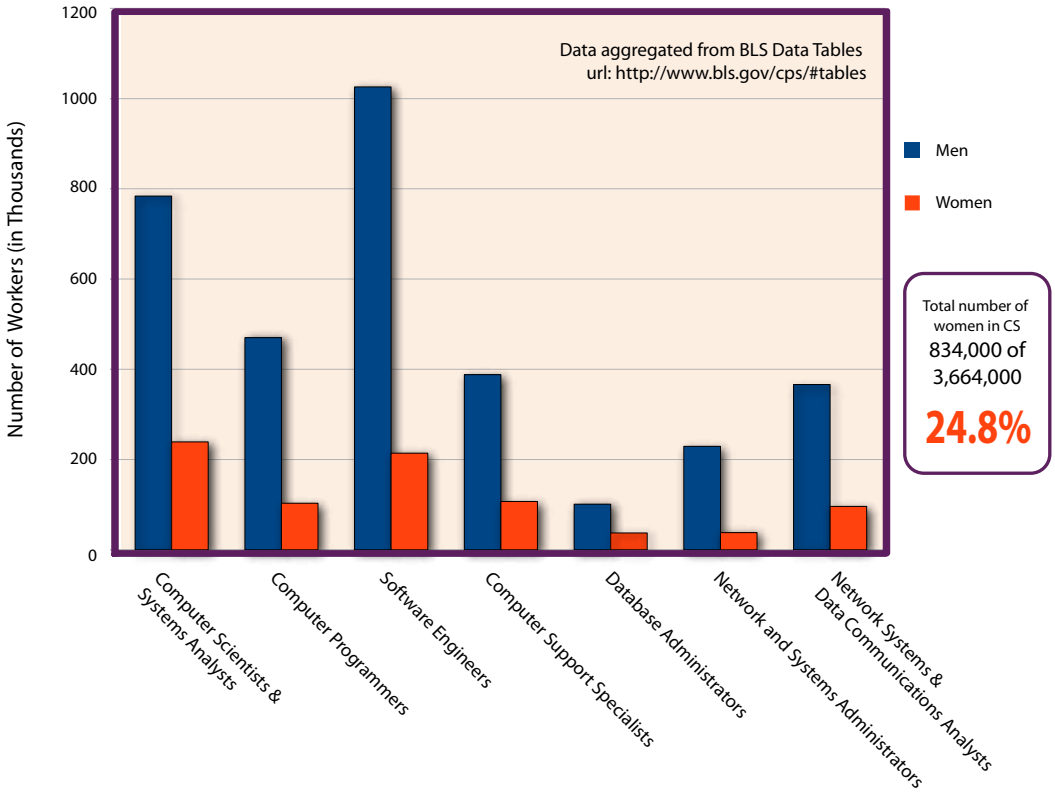
Much of the research on women in computer science focuses on their absence. Some hypotheses on related factors include sexism, preference for personal interaction, less access to resources as children, deficiency of female role models, and distaste for the stereotypes of geek culture. Lack of draw tends to be attributed to perception of the industry, while research on attrition points more toward working conditions and environments which cause women to feel like outsiders.

Before taking a look at the idea of gender in computer science, it is important in this day and age to recognize the sensitivity of categorizing "women" into one generic group. In this paper, the term "women" will be used as an abbreviation for "gender-schematic women" in accordance with the precedence set in "Evaluating Electronic Information Resources for Young Women: General Research Concepts" by D.E. Agosto (2000), which defines gender-schematic individuals as people who "view the world largely from a gendered point of view, bifurcating society into female and male components." Similarly, all other words which indicate generalizations on gender (such as "female" and "girls") will be used in the same vein.

Sexism is a topic which is frequently discussed in computer science, due in large part to the disparity between the number of men and women who pursue it. When analyzing an industry that is currently 75% male (see figure 3) the existence of bias is a reasonable inference. Unconscious bias, which often manifests itself without intentional ill-will by the offender, "may be exacerbated

in technical companies and departments", claims a 2009 report by the National Center for Women in Information Technology (Ashcraft and Blithe). The same study claims that women are more likely to stay quiet in meetings and not as likely to contribute their ideas when they feel unfavorably judged or devalued. This often leads to a high turnover rate due to lack of confidence and satisfaction.

Figure 3. Number of workers in computer science professions.



More blatant forms of discrimination are on the decline (Cesi and Williams 2011), but when perceived, they can play a large part in killing a woman's progress in computer science. Research performed on a group of graduate students shows that even when prejudice is not officially cited as a the main reason for leaving a CS program, a woman is 32 times more likely not

to complete her degree if she has previously thought of leaving due to sexism (Cohoon, Wu and Chao 2009).

Beyond bias, there are also gender-specific preferences which are likely to turn women away from technological professions; lack of collaboration is one such issue. Denise E. Agosto (2000) claims that girls learn through collaboration while boys learn through competition . This is especially intriguing at an undergraduate level where many of the initial courses focus on solitary work (Williams 2006). Such an introduction to the field reaffirms the belief that the life of a programmer is spent working alone — an idea that is both unappealing and frustrating to many women. In a 2010 study published by the Association for Psychological Science, STEM careers (Science, Technology, Engineering and Math) were statistically shown to be perceived as less communal in goals — that is less likely to be seen as working with or for the good of others — than alternative career choices.

A study by Jane Margolis and Allan Fisher (1997) found that many women want to use computers as a way to make society better. Accordingly, a survey done by ACM/WGBH in 2009 shows that the majority of girls prefer descriptions of computer science that appeal to their sense of community and ability to “do good” in the world, whereas boys prefer descriptions which showcas CS as a tool to help them be in control of their own lives.

Even so, current research suggests it is not only important how computers are presented to girls, but also when they are presented. In the paper, “Investigating the Incredible Shrinking Pipeline for Women in Computer Science” Camp and Gürer (1997) assert that access to computers and training in the concepts of computer science should be provided at preschool levels in order to give women the greatest chance to avoid developing insecurities about

their abilities. A survey conducted by Google in the Summer of 2010 confirms the importance of introducing computers early in life, finding that 98% of CS majors were exposed to CS before college, while only 48% of non-majors could say the same (Stephenson 2011).

Introducing women to computers when they are young is helpful, but not always quite enough. Unfortunately, in a mixed-gender education setting, boys will often appropriate the resources for themselves, leaving the girls out (Margolis and Fisher 2001). Similarly, teachers assume that boys are more interested in exploring computers, so they tend to select the boys over the girls when presenting computational opportunities (Kay 2007). In the book, "Unlocking the Clubhouse: Women in Computing," covering research done at the Carnegie Mellon School of Computer Science, Margolis and Fischer show that boys tend to have more access to computers at home. In fact, 40% of men (compared to just 17% of women) had been given their own computer as a child.

Very few girls are encouraged into computer science at home, and with a lack of female role models in the industry there is little to entice them from the inside. It is like the joke says, "Why aren't there very many women in computer science? Because there aren't very many women in computer science!" It sounds like a tautology, but really it's a self-preserving cycle that seems impossible to break. Fewer women in computer science means fewer role models for young girls. Fewer role models means less of a chance that girls will want to emulate the life of a computer scientist, which means there will continue to be fewer women in computer science.

A 1996 study by Gloria Townsend shows that just watching a brief video of female role models significantly improves female attitudes toward

computer science. Likewise, an article in *The Journal of Personality and Social Psychology* (Stout et al. 2011) displays the results of research on the effects of female role models among STEM students. The article indicates that exposure to female role models can not only prevent girls from developing negative attitudes toward the sciences, but also reverse negative perceptions that have already formed. The role model study shows that having male teachers will decrease a woman's self-efficacy (perception of her own capability) in STEM sciences over time and female teachers will increase a woman's self-efficacy over time.

Positive female examples go a long way toward improving a woman's attitude toward computer science, yet many women still feel out of place in an atmosphere of masculine geekery. Environmental cues alone (such as science fiction posters, soda cans and comic books) are enough to separate the girls from the boys when it comes to choosing a major. In an analysis of variance performed in 2009, Sapna Cheryan of University of Washington determined that a room adorned with items that were stereotypically associated with computer science affected women much more negatively than it did men. When polled about their interest in computer science, the women in a room with non-stereotypical decor were more than twice as likely to say they were interested in the major.

Environmental discomfort helps contribute to a woman's overall sense that she does not belong in a field where she has so little in common with the majority of her peers. Constant microbombardments of this nature tax the perseverance of an individual, causing one to lean heavily on her own belief in her abilities. Unfortunately, females in the computers science major tend to be less self-assured than male non-majors (Beyer, et al. 2003). This lack of

confidence in technical ability translates into stress and self-induced pressure (Abouseriea 1994), which contributes to heightened dissatisfaction.

Retaining women through college is difficult enough, but attrition in the workforce is considerable as well. Only 59% of women will stay in a technology career past their 10 year mark, as opposed to 83% of men (Ashcraft and Blithe 2009). In many cases, these women are not just quitting one specific job; they are quitting the field altogether. The most common factors contributing to the decision for women to leave are the same factors which prevent them from entering the workforce initially: sexism, isolation, and lack of female role models (Hewlett 2008).

Several entities have made attempts at remedies for this epidemic. Some countermeasures have proven more effective than others. Carnegie Mellon raised their percentage of women in the program to 34% by holding workshops for high school teachers and changing prerequisites to focus less on prior programming experience and more on ingenuity. They also made sure to provide multiple entry points into the major, so that students can be comfortable no matter what level of programming experience they are equipped with when they enter the department (Blum and Frieze 2005). Harvey Mudd College experienced similar success with a 2010 incoming class that was 52% female (Harvey Mudd College, 2011). Mudd attributes its increase in female participation to a technique similar to that of Carnegie Mellon. They also provide a plethora of research opportunities to students as young as sophomore level. At Harvey Mudd, female freshmen are invited to a computer science conference so they see the number of available role models from the beginning of their college experience (Sahil Luthra 2011).

CHAPTER IV

WHAT CAN BE DONE?

What can be done to fundamentally change the negative perception of computer science for women? A segment of the issues that turn women off from technological careers — including prevalent sexism and shortage of female role models — can be diminished by successfully bringing more women to the industry. Other matters, such as preference for personal interaction and distaste for the stereotypes of geek culture are deeply ingrained propensities that don't need to be changed in and of themselves as much as the perception that computer science is dissatisfying in those respects. Finally, the lack of access and opportunity to develop knowledge and engage young girls in computer science stands at the root of combating and preventing the barriers presented earlier in this paper.

Practice builds confidence, and confidence is an amazing ally for women in technology. When a woman is self-confident she is more able to stick up for her ideals, take risks and experience success in what she tries (Lynn 2008). Studies from as far back as 1979 (Lemkau) state that women who succeed in male-dominated careers tend be more confident than "the norms for women," a sentiment that is still echoed today (Eagly 2002). This kind of power is what a woman needs to fight the adversity described. The same 1979 study shows that women in male-dominated careers tend to have similar backgrounds, in that they were encouraged from a young age to explore both "masculine" and "feminine" opportunities. Open exploration, along with encouragement from a young age,

is precisely how computational thinking can help build confidence in the women of tomorrow.

Computational thinking combines abstraction and automation in problem solving. No firm standards have yet been adopted to easily define computational thinking, but the author has formulated a practical working definition by fusing together concepts presented by other authoritative resources (Wing 2006, Cooper et al. 2010, Google 2010) . One fundamental idea behind computational thinking is to break a problem up into its parts, abstracting out details so that patterns can be more easily seen. An algorithm is then constructed to automate the task of solving the problem itself. The point of computational thinking is not to give a precise number of steps to attempt in a particular sequence, but rather to train one's mind to recognize how a current issue might be similar to a problem that already has a known solution (Shekhar 2008). Often times, tweaking the solution to a similar problem can lead to a resolution for the unknown.

The understanding of computational thinking used in this paper goes one step past the descriptions from the resources mentioned above. As part of an attempt to make the need for computational thinking more clear, it has been further divided into two parts; the understanding of how to efficiently use computation, and the ability to prepare a problem for that computation.

A sorting problem will allow a straightforward example of computational thinking. Take, for instance, the need to pick the tallest item in a line-up. A human could simply glance at the list as a whole and instantly pick out the tallest specimen. A digital computer, on the other hand, does not have that capability. Heights would be stored as numbers in the machine memory. To find the largest height, each item would have to be compared to each other item in

order to be “sure” that the tallest is found. On the side of understanding efficient computation, one must know what the computer “needs” in order to solve a problem that may come naturally to a human mind. Computational thinking is enhanced by the ability to anticipate such hurdles introduced by the entity which will be performing the computation.

The second part of computational thinking is a directed form of analytical thinking. Using specific analytical methods, a problem can be cut into recognizable pieces, making it easier to work with. These pieces can often be formatted for computation before sewing the problem back together into a complete algorithm (list of instructions). Finally, the algorithm is translated into the language of the computing machine. That translation acts as the bridge between both halves of the computational thinking process. The bridge is often some form of code when using digital computers or electronics to solve a problem; but when the computing machine is a human brain, the bridge could take nearly any form — including a diagram, audio recording or a literal “list of instructions”.

Computational thinking is a tool that has been carefully developed as an addition to any field, not just fields which pertain to technology (Denning 2003, Perkovic et.al. 2010). The idea behind computational thinking is not just to prepare other disciplines for their inevitable blend with computer science, but to introduce other minds to the problem solving tools which are often taken for granted by computer scientists. Computational thinking can become a common mechanism for intellectuals from all areas of interest when looking at problems which are unsolved and unfamiliar.

Jeannette M. Wing, who has been referred to as the mother of computational thinking, declares “If we wanted to ensure a common and solid

basis of understanding and applying computational thinking for all, then this learning should best be done in the early years of childhood" (Wing 2008). Indeed, the precedent has been set for bringing useful skills to elementary school children in simplified forms. Counting and arithmetic are taught in preparation for more complicated mathematics education in the future, finger painting and color mixtures introduce art, and the alphabet is a precursor to reading and writing. Similarly, computational thinking is a particularly elegant way to introduce computer science at the K-5 grade level, due in large part to the fact that computers are not actually required. In much the same way as math is practiced with pencil and paper before a calculator is introduced, thinking computationally can be practiced through stories and puzzles well before a machine is necessary. By removing the technological requirement from the youngest age bracket, computational thinking — and therefore computer science — becomes more easily accessible to everyone.

CHAPTER V

THINKING MYSELF

Thinking Myself (http://games.thinkingmyself.com) is a website created by the author to engage girls in computational thinking as a step toward computer science. The environment is tailored to girls, using a colorful design with simple elements and several graphical components. Much of the user interface is based on intensive research surrounding the "rule/role" stage that happens between the third and fifth grade, where children start to read for the sake of learning and are more able to analyze longer, complicated sentences (Markoplous and Bekker 2003). Decisions for the overall look a re based on a study which finds that girls prefer simple layouts with a purple background and playful fonts (Taslim et.al. 2009).

Beyond the overall look, the feedback style is also deliberate and uplifting. Girls tend to respond very well to positive feedback. Negative feedback deters them much more severely than it seems to deter boys (Corpus and Lepper 2007) and is therefore avoided in almost every case. Whenever possible, incorrect answers are indicated by lack of positive signals rather than overt negative responses. To push the feedback scale even more into the affirmative, an applause button is located in the lower left-hand corner for affirmation on-demand.

Due to the female preference for eye contact and facial interaction (Baron-Cohen and Benenson 2003), a friendly guide, "Kiki" (see figure 4) , is introduced to walk the users through the lessons. Kiki is the representation of an amicable young girl who is dressed in feminine attire with bows in her

hair and a smile on her face. She is unintimidating, with proportions similar to those of a third grader. Kiki walks the students through each step of the tutorial, introducing them to new words, concepts and situations.

Figure 4. Screen shot from Thinking Myself, highlighting Kiki and the button structure.

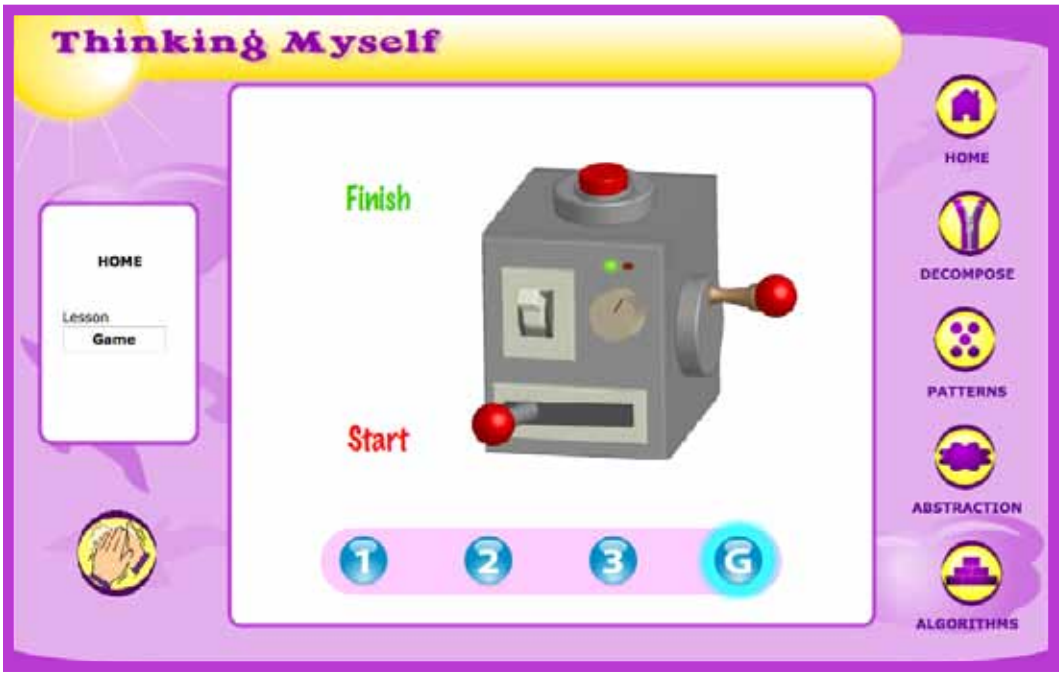
Thinking Myself consists of five sections - an introduction and four computational thinking lesson sections. The game advances automatically as each segment is passed, but a user has the option of jumping between sections using pictorial navigation buttons to the right of the main screen. Each lesson has multiple levels and a final game. Numbered, glowing buttons at the bottom of the game area give students a way to navigate between levels while playing.

5.1. Home: An Introduction

The first section, "Home", provides an introduction to the techniques that will be used to teach computational thinking. In the first level, Kiki introduces

herself and informs the users that she will be guiding them through lessons on how to solve problems without being explicitly told what to do. The expectation is set to allow third through fifth-graders the opportunity to understand that they can figure out solutions on their own, a concept which is illustrated using a game (see figure 5).

Figure 5. The Instruction-Free Machine



The "Home" game is a machine without directions. Knobs, handles and switches adorn the gadget without a single instruction on what to do. Only one small light provides feedback initially. In the beginning state, the light glows a beautiful green. If the user guesses the first correct move by pushing the red button on top, the light will stay green and a progress indicator will begin to rise on the left-hand side of the main window (see figure 7). Any other move at that time will either not elicit a response at all (such as clicking outside of the machine) or will cause the light to go red, indicating that something has been

clicked out of order (see figure 6). The machine behaves in a similar manner throughout the game; either displaying a green light and increasing the progress indicator or presenting the red light and resetting the progress indicator. Once each of the toggles have been clicked in the proper sequence, the progress bar reaches the top and Kiki comes back to tell the user that they have succeeded in winning the game. The user is then prompted to continue to the next lesson.

Figure 6. Machine and stage indicating incorrect move.

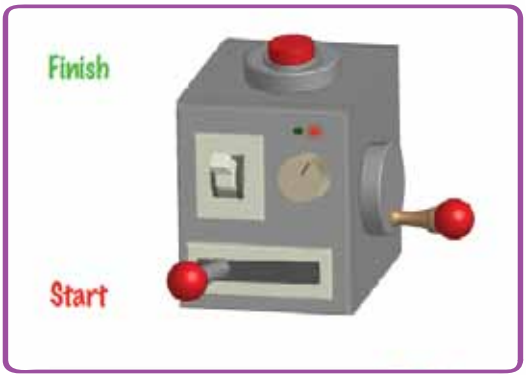
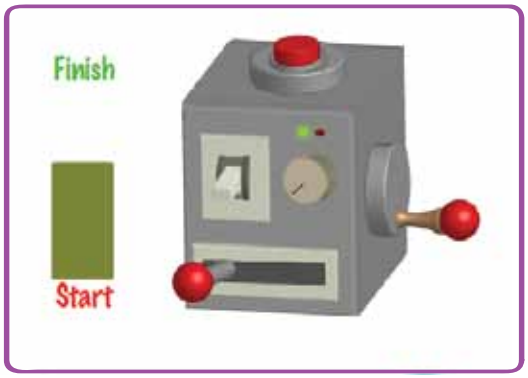


Figure 7. Machine and stage indicating progress.



Once the user has been put in the figure-it-out state of mind, the formal computational thinking lessons begin. Thinking Myself does not hide complex ideas behind amusing activities, it uses those activities to highlight the complex ideas. The user is told definitively that they will be experiencing "computational thinking". Each section advertises the word for which it is themed, even though those themes are technical and multi-syllabic. The idea is to introduce children to formal concepts in a fun and engaging manner, taking the sting out of words like "decomposition" and "algorithm".

5.2. Lesson 1: Decompose

The first official lesson is under the button “Decompose”. Decomposition is defined in this lesson as “taking large, difficult problems and breaking them down into smaller, easier ones”. All of the concepts chosen for Thinking Myself were modified to be easily understood while remaining true to the computer scientist’s interpretation of the notion.

In “Decompose”, Kiki walks the user through an example of decomposition that he or she may already be familiar with; showing how multiplication can be broken down into addition. She then goes on to a more complicated example which is most-likely entirely new to the student. After Kiki shows how to count the faces on a cube made of smaller cubes by taking it apart, she leads the user into another game. Here, the user is presented with several numbers to sort. As pictured in figure 8, Kiki guides the student through the sorting process by decomposing the task into three smaller ones using the classic “bucket sort” method.

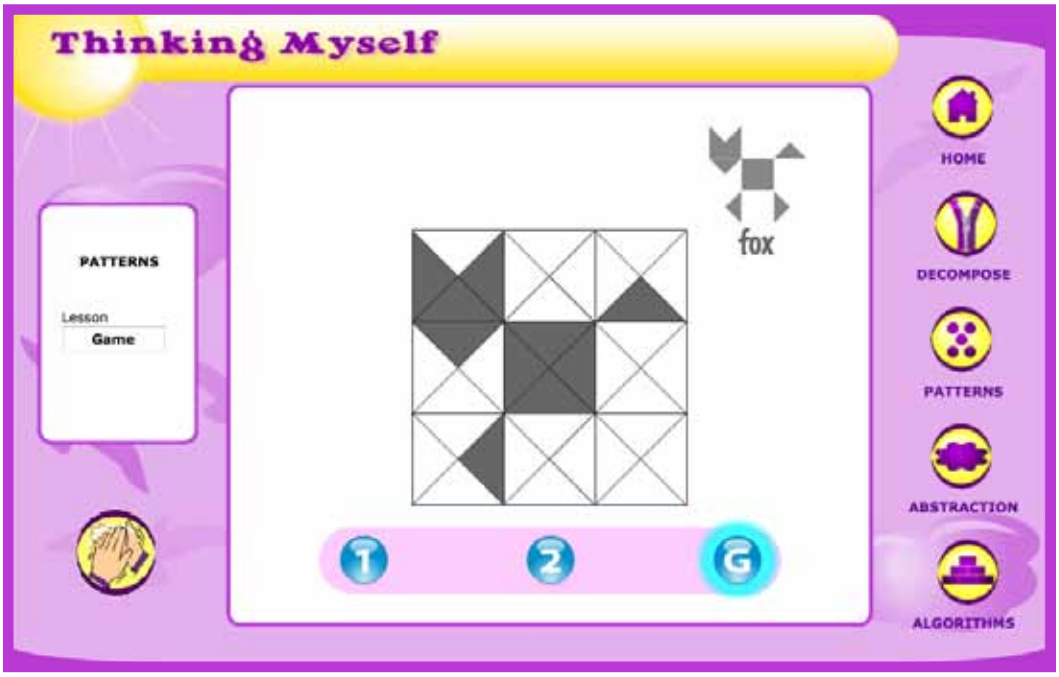
Figure 8. Bucket Sort game at the end of the “Decompose” lesson.

The game is made positive by choosing to not allow incorrect moves rather than provide negative feedback such as buzzing noises or phrases like "Wrong Move." If a student tries to drop a number into an incorrect bucket, the number will simply reset itself to the top of the screen, leading the user to conclude that no progress can be made by continuing to attempt that action. Once the sorting is completed, Kiki praises the user and reinforces the idea that some problems are made easier by breaking them up into smaller pieces.

5.3. Lesson 2: Patterns

Once "Decompose" is completed, the student is guided into "Patterns". A pattern is defined as "repetition in design" or "similar qualities that are shared by a number of different items." This explanation is sufficient for the needs of Thinking Myself, but should not present itself as contrary to what students have already learned in previous grades. Kiki uses short games to walk the student through different ways of looking at and thinking about patterns. Once she has displayed both "repetition in design" and "similar qualities that are shared by a number of different items," it's time for another end-of-lesson game. This game, pictured in figure 9, is loosely modeled after tangrams, encouraging kids to look for patterns in a sea of blank triangles. After successfully spotting and recreating the patterns, the user is given accolades and encouraged on to the next lesson.

Figure 9. The Triangle Game at the end of the "Patterns" section.

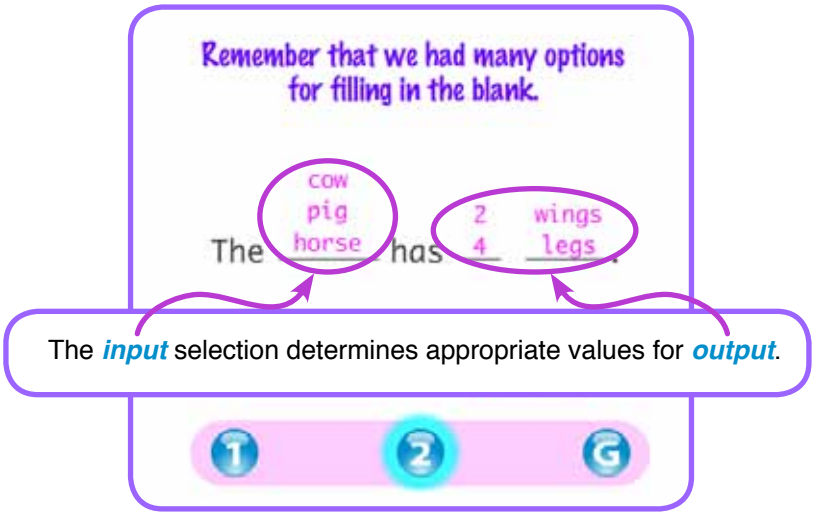


5.4. Lesson 3: Abstraction

"Abstraction" is presented as the "art of taking the details out of a problem so that you can make a solution work for many different things." This definition, while simple in concept, is actually more strict than the common definition. By adding the purpose of abstraction to the definition, the user is more easily guided to the object of the following example. In that example, the student is shown three similar sentences and asked to determine which words are identical and which words are different. By collapsing the sentences onto one another, it becomes obvious that some words line up perfectly and others become a jumble of letters. The words that jumble are then blacked out and turned into a "blank". The act of leaving the static words visible and creating a blank for the changing word is reinforced as abstraction.

As a next step, the sentence is abstracted even further by identifying other parts of the line that could change under different circumstances (see figure 10). After repeating this idea, Kiki comes back to help the student understand the use of abstraction. She defines the parts of the sentence that can change as *variables*. Variables are then used to introduce *inputs*; where an input is the piece of a sentence that determines how other abstracted blanks get filled-in. Similarly, *outputs* are defined as the values that are appropriate for the selected input.

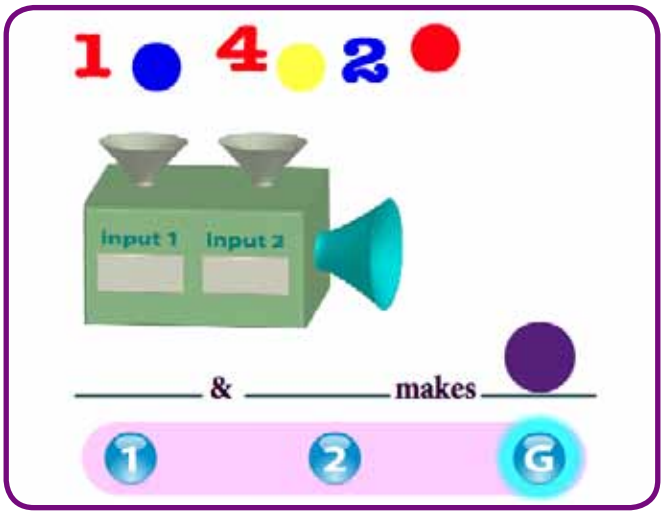
Figure 10. Input/output relationship for abstracted sentence



Fresh from the tutorial on variables, inputs, and outputs, students are led into another game. This game is a machine that combines two inputs into one output. The user is given several variables to choose from and is asked to select the two that will reproduce the output displayed in the bottom right-hand corner (see figure 11). Again using the concept of positive-feedback only, the machine disallows incorrect moves rather than recognizing them with buzzing or negative responses. If an incorrect selection is made, the machine will not receive the variable and it will be returned to the selection area. On the other

hand, if a correct input is placed in the machine, signs of progress will begin immediately.

Figure 11. Two input machine



Once the correct variables are dropped into the contraption, it begins to shimmy and shake while the "input" fields populate with the selected variable names. After the anticipation-building action phase, the machine stops and spits out the output created by blending the two selected inputs. The output lines up with the sample in the bottom right-hand corner and success is proclaimed in the form of congratulatory affirmation.

5.5. Lesson 4: Algorithms

Finally, the user is presented with the lesson on "Algorithms". The word may sound intimidating, but Kiki simplifies the idea by defining an algorithm as a set of instructions for completing a task. The first lesson compares an algorithm to a recipe and uses that analogy to introduce loops. By following a numbered list of steps, it becomes obvious when one step is repeated multiple

times. After that illustration, the second level takes a brief peek into algorithm efficiency. Continuing with the recipe scenario, Thinking Myself uses an interactive animation to show that the amount of time to complete two batches of cookies can vary based on the algorithm used. It takes less time, for example, to use each ingredient once and double the amount added, than it would to add each ingredient in sequence two times.

The final game, displayed in figure 12, tests the student's ability to follow algorithms. Disguised as a treasure hunt, the user is given instructions to follow one at a time, each step leading them closer to finding the treasure. If all of the instructions are followed correctly, the user will land on a specific island, which will play the animation of a growing treasure chest. If the user does not end on the proper island, the game will go back to the first instruction and guide them through another time. When the treasure is found, Kiki reappears to praise the students and wrap up the lesson, giving them the chance to play again from the beginning if they so desire.

Figure 12. Algorithm treasure hunt

5.6. The Game in Use

In practice, Thinking Myself has created some interesting observations. Initial testing was done with nine adults, observed individually, each with some background in STEM. Perhaps counter to intuition, it was the adults with the most experience in their fields who took the longest to solve the Instruction-Free Machine. The tool was also introduced to a six-year-old boy who solved the Instruction-Free Machine quite quickly, but clicked through the majority of the lessons without reading the text just to get from game to game. His brother, a seven-year-old also mastered the end games, but he showed more patience for the lessons and animations which tied the ideas together.

The program was presented at the end of a local girls' science camp as an additional activity for those who had completed their pair-programming exercise. Every one of the sixteen girls made sure that they had time to play Thinking Myself. Each of them was drawn into the lessons, discussing what they saw with their partner and giggling as they initiated applause for themselves time and again with the animated clapping button.

The experience appeared to be very different for the adults and the children. Most notably, children appeared to be much quicker at completing lessons and solving the games than adults were. Including the six-year old, every child was faster than every adult at solving the machine without instructions in the introductory session. The fastest time for solving the Instruction-Free Machine belonged to a fourth-grade girl, followed closely by several other fourth and fifth-graders. The longest time belongs to a sixty-year-old engineer.

Other observations worth mentioning were the attitudes of the participants. In every case, the users were not told what to expect and started

with some uncertainty. Inevitably, by the middle of the game, users were smiling, giggling or cheering for themselves. Thinking Myself did not appear to easily cause frustration in any of the participants observed. Since there are no time limits, wrong answers or negative sounds, shame and embarrassment are not anticipated to be common byproducts of the tool.

5.7. The Game in Analog

Thinking Myself is a proof of concept that computational thinking can be made entertaining and accessible for grade school students. For those who have access to the Internet, Thinking Myself provides a self-contained lesson plan, which can be covered in one or two days worth of lab time. Without computers, Thinking Myself can be replicated with active instruction and creative game play in much the same manner as it is presented online. Fortunately, the lesson portions of Thinking Myself are easy to replicate in a classroom environment with a combination of verbal explanation and tactile versions of the examples given online. The games are only slightly harder to simulate.

The most difficult of the games to reproduce is the Instruction-Free Machine. Any number of creative games can replace the purpose of the machine as long as the instructor gives no instructions, hints or demonstrations. The most effective substitutes will embed automatic cues without negative-feedback.

5.8. Instructionless Offline

Teachers are invited to visit games.thinkingmyself.com/instructionless.pdf to download the Instruction-Free Game. Inside the PDF is a page of cards which can be printed out on full sheets of paper. The cards form a series of math problems, with each card building on what has been done so far. Each

page is placed face-down in a 5x5 grid. Students will intuitively want to turn them over. The printed side will guide the next move, though the pattern is far from obvious. Each card will suggest the next when played correctly. The purpose of the paper game is analogous to that of the online version. As long as the children are allowed to fully explore the task with no “help”, they will receive the priming necessary to open their minds to the exploratory nature of computational thinking.

5.9. Sorting Offline

The sorting game at the end of “Decompose” is quite simple to replicate without digital assistance. A teacher can start with numbers from 1-15 written on individual sheets of paper, then mix them up in a big pile on the ground. Using three buckets or boxes (which have been labeled with the intervals 1-5, 6-10 & 11-15) students can pick numbers one by one and put them in the appropriate bin. Once all numbers have been placed in their correct locations, the buckets can be unloaded, one at a time, and easily sorted by sight. Bucket Sort is a great game to revisit after finishing the “Algorithms” section, in order to discuss methods of efficiently sorting the contents of the individual bins. An array of numbers is located at games.thinkingmyself.com/BucketSort.pdf for download.

5.10. Triangle Puzzle Offline

Next in the series is the Triangle Puzzle from the “Patterns” section. A physical copy can be printed at games.thinkingmyself.com/TriangleFox.pdf or games.thinkingmyself.com/TriangleBoat.pdf for classroom use. The pages contain both the sample image and the grid of triangles for completing the puzzle. An ambitious teacher could provide students with laminated triangles

to arrange on the printed template, but coloring triangles with crayons is also effective. The purpose of this game is to spot a pattern in a field of apparent chaos. Any method of highlighting that pattern is sufficient.

5.11. Dual-Input Machine Offline

Under the "Abstraction" heading is the Dual-Input Machine. The intention behind this machine is to show that when the details of specific variables are abstracted out, what's left is a versatile solution that can be applied to many types of inputs. A fun variation to play in the classroom includes various objects and a large box or curtain for the instructor to hide behind. This version, called "Guess What the Box Does", allows children to plainly see the results of a mystery algorithm on a set of inputs. The task is for the students to guess what the machine is programmed to do based on what outputs are produced by different combinations of inputs. An example of game play could include a pile of paper animals on a table. Students would line up to introduce inputs to the machine. Say, in this round, the machine required only one input. A student would choose a paper animal and "drop it into the machine" by handing it to the teacher behind the curtain. The teacher could make machine noises and hand the output to the student on the other side, the output in this case being three copies of the selected animal. It would not take long for students to determine that the machine was tripling each of the input animals. More complicated versions include: two-animal inputs where the output is the head of one and the tail of the other, two color inputs where the output is a blend of the colors, or three number inputs where the output becomes the sum of the numbers. An instructor could certainly get even more abstract by

allowing colors, numbers and animals together, then using a solution which can incorporate blends of the variables when necessary.

5.12. Island Game Offline

Finally, an offline version of the game at the end of the "Algorithms" section is provided at games.thinkingmyself.com/IslandGame.pdf for printing. This game consists of a blue island and several brown islands, one of which has a treasure printed on the opposite side. The instructor can place the islands in a grid or any other pattern which suits their needs. Starting at the blue island and ending at the island with the treasure beneath, the instructor can create a hunt using as many steps as desired to get from one to the other. Instructions can be given orally or written down in advance and handed to the students. After the game has been played successfully, a variation can be introduced where the students are asked to come up with their own algorithm to get from the same beginning point to the same ending point in a different way. Challenges can be issued including "Find the Shortest Path" and "Find the Longest Path". This is an exciting, hands-on way to learn that not all algorithms are equal.

5.13. Summary

While the online version of Thinking Myself requires less paper and planning, the offline methods are helpful when it comes to supplementing or substituting for the digital version if teachers encounter a lack of Internet access. Of course, one of the most important pieces to Thinking Myself is the use of actual computational thinking terminology as a way to help students connect intimidating words with fun ideas. For that reason - whatever the method - it is important for instructors to pepper their lessons with words like *abstraction*,

algorithm, *decomposition*, *variables*, *input* and *output* in order to inspire confidence.

Large companies, such as Google and Intel, are making an effort to bring computational thinking to students. Almost all of their effort has been focused on grades 6-16. For maximum effect on women in computer science, that technological confidence and esteem must be built before girls reach middle school and many are irreparably turned off from STEM subjects (Lanzer 2009). Some helpful resources similar to the offline concepts of Thinking Myself which are appropriate for elementary school exist, covering additional topics in computer science that can be related back to the concepts in this thesis. One such resource is Computer Science Unplugged. Teachers who are interested in fortifying their computational thinking lessons by blending them with practical application can find a wealth of activities on the Computer Science Unplugged website (http://csunplugged.org).

CHAPTER VI

CONCLUSION

The retention of women in computer science among those who were introduced to technology prior to college is superior to that of women who were not. In an attempt to begin building a foundation for women as young as possible, this thesis proposes computational thinking as a viable starting place. Computational thinking has already been accepted as preliminary concept for blending technology into seemingly unrelated professions and its potential in developing minds is exciting. Before computers are ever introduced into a classroom, computational thinking can begin to prepare girls for the skills that they will need to acquire in order to be successful in a world that is increasingly dependent on technology.

This thesis also presented Thinking Myself, a visually tailored, child-friendly application which was designed to heighten the interest of girls in computer science through fun and challenging examples of computational thinking. Intentionally sparse on instructions, Thinking Myself encourages trial and error, allowing students to figure out answers by themselves, then it rewards them with praise when they succeed. Provided as a cost-free classroom tool, Thinking Myself can be used online or pieced out as offline activities which can be incorporated and easily understood by children with at least a third-grade education.

As a growing program, Thinking Myself could benefit from several additional areas of research. Longitudinal data is not yet available for the effects of computational thinking on problem solving ability, but strengths

and weaknesses of a child trained to think computationally would certainly be useful when selecting the level of detail to cover early in the elementary grades. Additionally, a strong correlation between the age of introduction to computational thinking, length of exposure to computational thinking lessons, and self-assessed technological interest beyond middle-school is likely to inspire the adoption of more advanced computer science curriculum prior to collegiate education.

When analyzing the effectiveness of Thinking Myself as tool, research can be done on all aspects of the program, from enjoyment of play at different age-levels to analysis on a student's ability to solve various problems before and after exposure to the lessons. The project was constructed in a relatively modular manner, allowing for an increased number of lessons in each section or even additional topics. The framework was created to allow for a database where a login feature can be included so that each student's progress can be securely tracked. The content can easily be enriched, at any point, with new animations, interactive quizzes and video tutorials. Future collaboration toward the growth of this program is welcomed in an effort to continue building strength among young women in computer science.

REFERENCES CITED

Abouserie, Reda. 1994. Sources and levels of stress in relation to locus of control and self esteem in university students. *Educational Psychology*. 14, no. 3: 323-30.

Abraham, L. B., M. P. Mörn, and A. Vollman. 2010. Women on the web: How women are shaping the Internet. url: http://www.comscore.com/Press_Events/Presentations_Whitepapers/2010/Women_on_the_Web_How_Women_are_Shaping_the_Internet (accessed April 01, 2011)

ACM, WGBH. 2009. Interim report. New Image for Computing. url: www.acm.org/membership/NIC.pdf (accessed April 15, 2011.)

Agosto, D. E. 2000 Evaluating electronic information resources for young women: General research concepts. url: http://girlstech.douglass.rutgers.edu/PDF/completereport%20.pdf (accessed April 01, 2011)

Ashcraft, C., and S. Blithe. 2009. Women in IT: The facts. Boulder, CO: National Center for Women and Information Technology.

Astrachan, O., S. Hambrusch, J. Peckham, and A. Settle. 2009. The Present and future of computational thinking. *SIGCSE BULLETIN*. 41, no. 1: 549-550.

Baron-Cohen, Simon, and Joyce F Benenson. 2003. Essential difference: Men, women and the extreme male brain. *Nature*. 424, no. 6945: 132.

Barr, D., J. Harrison, and L. Conery, 2011. Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*. 38, no. 6: 20-23.

Beyer, Sylvia, Kristina Rynes, Julie Perrault, Kelly Hay, Susan Haller, Gender differences in computer science students. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, February 19-23, 2003, Reno, Navada, USA [doi>10.1145/ 611892.611930]

Blum, L. and C Frieze. 2005. As the culture of computing evolves, Similarity can be the difference. *Frontiers*. 26, no. 1.

Bureau of Labor Statistics, Current Population Survey. 2010. Table 11: Employed persons by detailed occupation, sex, race, and Hispanic or Latino ethnicity - Annual averages.

Camp, T., and D. Gürer. 1997. Investigating the Incredible Shrinking Pipeline for Women in Computer Science. *ACM Committee on Women in Computing*.

Ceci S.J., and W.M. Williams. 2011. Understanding current causes of women's underrepresentation in science. Proceedings of the National Academy of Sciences of the United States of America. 108, no. 8: 3157-3162.

Cheryan, S., V.C. Plaut, P.G. Davies, and C.M. Steele. 2009. Ambient belonging: How stereotypical cues impact gender participation in computer science. *Journal of Personality and Social Psychology*. 97, no. 6: 1045-1060.

Cohoon, J.M., Z. Wu, and J. Chao. 2009. Sexism: toxic to women's persistence in CSE doctoral programs. *SIGCSE BULLETIN*. 41, no. 1: 158-162.

Cooper, S., L.C. Perez, and D. Rainey. 2010. Education: K-12 computational learning. *Communications of the ACM*. 53, no. 11: 27-29.

Corpus, Jennifer Henderlong, and Mark R. Lepper. 2007. The effects of person versus performance praise on children's motivation: Gender and age as moderating factors. *Educational Psychology*. 27, no. 4: 487-508.

Denning, P.J. 2010. The great principles of computing. *American Scientist*. 98, no. 5: 369-372.

Diekman, A.B., E.R. Brown, A.M. Johnston, and E.K. Clark. 2010. Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers. *Psychological Science.* 21, no. 8: 1051-1057.

Dreze, X, and F. Zufryden. 1999. Is internet advertising ready for prime time? *Communication Abstracts*. 22, no. 2.

Eagly, A.H., and S.J. Karau. 2002. Role congruity theory of prejudice toward female leaders. *Psychological Review*. 109, no. 3: 573-98.

Farrington, Gregory. 1996. ENIAC: The birth of the information age. *Popular Science*. Vol. 248, no. 3: 74-76.

Fuegi, John, and Jo Francis. 2003. Lovelace & Babbage and the creation of the 1843 'Notes'. *IEEE Annals of the History of Computing*. 25, no. 4: 16.

Goldstine, Herman H. 1972. *The computer: from Pascal to von Neumann.* Princeton, New Jersey: Princeton University Press. ISBN 0-691-02367-0.

Goodnight G.T., and S. Green. 2010. Rhetoric, risk, and markets: The dot-com bubble. *Quarterly Journal of Speech*. 96, no. 2: 115-140.

Google. 2011. Exploring computational thinking. url: http://www.google.com/edu/computational-thinking/what-is-ct.html (accessed March 30, 2011.)

Harvey Mudd College. 2011. Incoming class of 2014 the most diverse yet. url: http://www.hmc.edu/specialinterestfeatures/oncampus/class-of-2014-most-diverse-yet.html (accessed May 02, 2011.)

Information Solutions Group. 2010. Social Gaming Research. url : http://www.infosolutionsgroup.com/2010_PopCap_Social_Gaming_Research_Results.pdf (accessed April 21, 2011).

Joy, L., N.M. Carter, H.M. Wagner and S. Narayanan. 2007. The bottom line: Corporate performance and women's representation on boards. *New York: Catalyst.*

Hewlett, Sylvia, Buck Luce, Lisa Servon, Laura Sherbin, Eytan Sosnovich, Karen Sumberg. 2008. *The Athena factor: Reversing the brain drain in science, engineering, and technology. Harvard Business Publishing.*

Kay, R. H. 2007. Gender differences in computer attitudes, ability, and use in the elementary classroom. *Research into Practice, Ontario Ministry of Education*. Monograph #8, 1-4.

Klawe, M., and N. Levenson. 1995. Women in computing: Where are we now? *COMMUNICATIONS- ACM*. 38, no. 1: 29.

Lanzer, F. 2009. Attracting girls to engineering & technology: Reach them before they're turned off. *Proceedings of the 2009 Mid-Atlantic Section Conference of the American Society for Engineering Education*. Baltimore, MD. url: http://ola3.aacc.edu/fplanzer/documents (accessed March 28, 2011.)

Lemkau, Jeanne Parr. 1980. Personality and background characteristics of women in male-dominated occupations: A review. *Psychology of Women Quarterly*. 4, no. 2: 221-39.

Lynn, Adele B. *The EQ interview finding employees with high emotional intelligence.* New York: AMACOM, 2008. <http://public.eblib.com/EBLPublic/PublicView.do?ptiID=408794>.

Margolis, Jane and Allan Fisher. 1997. Geek mythology and attracting undergraduate women to computer science. Impacting change through collaboration. *Proceedings of the Joint National Conference of the Women in Engineering Program Advocates Network and the National Association of Minority Engineering Program Administrators.*

Margolis, Jane, Allan Fisher, and Faye Miller. 2000. INCLUSIONS AND EXCLUSIONS: GENDER DIFFERENCES AND DIVERSITY AMONG WOMEN - The anatomy of interest: Women in undergraduate computer science. *Women's Studies Quarterly*. 28, no. 1: 104.

Margolis, Jane, and Allan Fisher. 2002. *Unlocking the clubhouse: Women in computing*. Cambridge, Mass: MIT Press.

Markopoulos, P., and M. Bekker. 2003. Interaction design and children. *Interacting with Computers*. 15, no. 2: 141-149.

Moursund, David G. 2006. Computational thinking and math maturity improving math education in K-8 schools. Eugene, OR. David Moursund.

National Science Foundation, Division of Science Resources Statistics. 2008. Science and engineering degrees: 1966–2006. *Detailed Statistical Tables NSF 08-321*. Arlington, VA. url: http://www.nsf.gov/statistics/nsf08321 (accessed March 13, 2011).

Perkovic, L., A. Settle, S. Hwang and J. Jones. 2010. A framework for computational thinking across the curriculum. *Proceedings of the 2010 Conference on Innovation and Technology in Computer Science Education*. 123--127.

Rice, John R, and Saul Rosen. 2004. Computer sciences at Purdue University -- 1962 to 2000. *IEEE Annals of the History of Computing*. 26, no. 2: 48.

Schenkel, Susan. 1984. *Giving away success: Why women "get stuck" and what to do about it*. New York: McGraw-Hill.

Shekhar, Shashi, and Hui Xiong. 2008. *Encyclopedia of GIS*. New York: Springer, url: http://dx.doi.org/10.1007/978-0-387-35973-1 (accessed April 24, 2011.)

Stephenson, Chris. 2011. No more excuses for lack of access. *CSTA: The Advocate.* url:http://blog.acm.org/csta (accessed March 30, 2011.)

Stout, J.G., N. Dasgupta, M Hunsinger, and MA McManus. 2011. STEMing the tide: Using ingroup experts to inoculate women's self-concept in science, technology, engineering, and mathematics (STEM). J*ournal of Personality and Social Psychology*. 100, no. 2: 255-70.

Taslim, J., Wan Adnan, and N.A. Abu Bakar. 2009. Investigating children preferences of a user interface design. *Lecture Notes in Computer Science*. 5610 LNCS, no. PART 1: 510-513.

Luthra, Sahil. 2011. The Brown Daily Herald. url: http://www.browndailyherald.com (accessed April 24, 2011.)

Towsend, G. C. 1996. Viewing video-taped role models improves female attitudes toward computer science. *SIGCSE BULLETIN*. 28, no. 1: 42-46.

US Census Bureau: State and County Quick Facts. 2009. USA quick facts. url: http://quickfacts.census.gov/qfd/states/00000.html (accessed May 07, 2011).

US Department of Education, National Center for Education Statistics. 2009. Department of Education tables and figures. url:http://nces.ed.gov/programs/digest/d09/tables/dt09_303.asp (accessed March 12, 2011).

U.S. Department of Labor, Bureau of Labor Statistics, 2009. Employment and earnings, 2009 annual averages and the monthly labor review. url: http://www.dol.gov/wb/stats/ (accessed April 14, 2011)

Wilford, John N. 2008. Discovering how Greeks computed in 100 B.C. *The New York Times* [New York, New York] 31 July 2008. Print.

Williams, Laurie. 2006. SERIES - Broadening participation in computing - Debunking the nerd stereotype with pair programming. *Computer*. 39, no. 5: 83.

Wing, Jeannette M. 2006. Viewpoint - Computational thinking. *Communications of the ACM*. 49, no. 3: 33.

Wing, Jeannette M. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 366, no. 1881: 3717-3725.