Computing Attitudes: Will Teaching 2nd Grade Students Computer Science Improve their Self-Efficacy and Attitude and Eliminate Gender Gaps?

Amanda Kind
St. Mary’s College of Maryland

Abstract: Many students are not being introduced to the subjects of computer science (CS) and computer programming until high school or college and there is an underrepresentation of women in all fields of computer science. This study focused on teaching early elementary students (2nd-graders) computer programming using the developmentally appropriate language called ScratchJr and implementing it into the science curriculum through storytelling. It was expected that teaching early elementary students computer programming would increase students’ self-efficacy and attitude towards computers and CS, especially girls. It was found that 2nd-grade student’s skills in CS significantly increased. In addition, teaching computer programming increased students’ self-efficacy and attitude towards CS. While my sample size wasn’t big enough to draw definitive, significant conclusions, the trend was for girls’ self-efficacy and attitudes towards computers to be lower than boys’ at the start of the project, but after the intervention girls’ and boys’ scores were similar.

Introduction

Women are currently underrepresented in computer science (Zweben, 2013). Studies have shown that middle school is a critical age, during which many girls turn away from scientific and mathematical pursuits, including computer science (AAUW). Students start forming attitudes about different subjects during the early elementary years. Andre, Whigham, Hendrickson, and Chambers (1999) found that in grades as early as K-3, students have already begun to stereotype jobs by sex-role. By the age of five, both boys and girls have already begun to stereotype jobs that relate to math and sciences as male professions. By the time students reach grades 4-5, girls are already more likely than boys to prefer reading to physical science. In addition, girls are less likely to enroll in computer science classes in their undergraduate education than boys (Zweben, 2013).

Along with an underrepresentation of girls in computer science, there is an overall lack of computer science majors in general. According to the Congressional Research Service, as of 2012, computer occupations accounted for 55.2% of all science and engineering jobs (Sargent, 2014). In the next 10 years, they project that 68.3% of all science and engineering jobs will be in computer science and engineering, or more generally,
information technology. Yet, the U.S. Bureau of Labor Statistics projects that in the next decade there will be about 1 million more U.S. jobs in the tech sector than computer science graduates to fill them. In order to remedy this situation it is imperative that we train the next generation of computer scientists and engineers and get them started early, in particular, in elementary school and middle school (Tucker, Deek, Jones, McCowan, Stephenson, & Verno, 2003). Teaching students in early elementary school could not only possibly change girls stereotype towards computer sciences, but also generate more interest in and exposure to the computer sciences in general. Thus perhaps making both boys and girls more likely to pursue computer sciences as a career path in the future.

A potential reason for this discrepancy could be due to the fact that students start thinking about their future careers while still in middle school. Middle schoolers have a good idea of careers in education and the medical field as they have personal experience with teachers and doctors. However, according to the College Board (2014) it is estimated that only about 10 percent of K-12 schools teach computer science. Therefore, the majority of middle schoolers have little exposure or knowledge of careers in computer science other than the “media images of the noninteractive geek (most likely male) sitting in his cubicle all day” (Rodger, Hayes, Lezin, Qin, Nelson, & Tucker, 2009). Thus, by giving all students, especially girls, a positive first programming experience early on in life—in elementary school—we may be able to increase girls’, and all students’, participation in Computer Science.

To date there has been successful research on teaching younger students computer science. However, this research has mostly focused on students in the upper elementary years and middle school. As I will discuss in my action, there has been very little research that approached teaching computer science in the early elementary years and even fewer studies that actually integrate computer programming into the curriculum. In addition, most programming languages written for students, such as Alice and Scratch, are not intended for young children. Some programming tools target children at least 7-8 years old, but most target typically older students (Flannery, Silverman, Kazakoff, Bers, Bont, & Resnick, 2013), making them not developmentally appropriate for students in grades K-2. However, just like students need practice with basic math concepts early on before they can graduate to more difficult constructs, I believe children need practice with basic programming concepts early on in order to understand more advanced concepts that rely on the basics as foundation. Just like in mathematics courses, students need time working slowly with basic constructs in order to fully understand and comprehend the later, more advanced concepts. Therefore, it is important that we start teaching computer science in the early years of elementary school in order to expose students and gain their interest in computer sciences as a future career path, especially as it is a rising career (Congressional Research Service, 2012).

The purpose of my study was to explore the feasibility of teaching computer science in the early elementary classroom. I sought to determine if teaching computer science to early elementary (K-2) students is effective, and to examine if there are gender effects in computer science on self-efficacy, attitude, and confidence for early elementary students. Specifically, my study focused on answering these questions: (a) Is there a difference between girls and boys self-efficacy and attitudes towards computers for early elementary students?; (b) Does teaching elementary students computer programming increase students self-efficacy and attitude toward computers and computer science?; and (c) Does
understanding and success in learning computer programming affect their self-efficacy and attitudes in computers and computer science? In the next section I will justify the reason we need to teach computer science in the early elementary schools. I will start out by explaining how teaching computer science in elementary school is good for all students, especially girls. I will then go into reasons why teaching computer science in the early years could also be beneficial for the nation as a whole. Finally, I will end with why my research would be beneficial and adds to the knowledge gap, as well as present existing research. The findings from this study are of interest to teachers and administrators.

**Literature review/theoretical framework**

**Why Teach Early Elementary Students Computer Science?**

Fifteen years ago the National Research Council (1999) defined an idea called Information Technology (IT) fluency emphasizing students’ capability of independently learning and using a new technology as it evolves throughout their lifetime. One important aspect of IT fluency is its emphasis on computer programming as medium of learning algorithmic thinking, which involves decomposing a problem into a sequence of steps and precisely identifying the steps so that they can be solved efficiently (National Research Council, 1999). Computer programming incorporates advanced algorithmic thinking, which cannot be taught through elementary mathematics alone. However, in the younger grades, computer programming is not generally taught. Therefore students miss out on the opportunity to learn advanced algorithmic thinking skills.

Originally, this oversight could be attributed to the fact that learning the syntax of text-based programming languages was too difficult for young children. In recent years, this discrepancy has been remedied with the creation of developmentally appropriate programming languages such as Logo (Fessakis, Gouli, & Mavroudi, 2013), Alice (Cooper, Dann, & Pausch, 2000), Scratch (Maloney et al., 2010), and even more recently ScratchJr (Flannery et al., 2013). With the development of these new programming languages, researchers found that students as young as five or six were capable of learning most of the basic programming concepts with sufficient support from a teacher or adult (Adams & Webster, 2012; Baytak & Land, 2011; Kelleher, Pausch, & Kiesler, 2007; Lee, 2010).

Through these developmentally appropriate programming languages, students are capable of understanding and grasping basic computer programming concepts. Fessakis, Gouli, and Mavroudi (2013), in a recent case study, created a developmentally appropriate Logo-programming environment—the Ladybug leaf and the Ladybug maze—in which 5-6 year old students had the opportunity to develop mathematical skills (counting, number comparison, angle turn concepts) and were introduced to basic computer programming concepts (command, sequential execution of commands, program, logical errors, testing and debugging of programs).

They found that not only were students capable of solving problems, students actively and intensively solved the problems posed and asked to do more activities even after engaging in problem solving for over an hour. Thus, suggesting that students enjoyed programming and are motivated to continue learning, given the right situation. Bromwich, Masoodian, and Rogers (2012) confirm this, finding that when asked to program an avatar to navigate a maze to reach the finish, middle school students were engaged with the game.
for the full length of the time and were motivated to continue learning programming. However, gaming projects such as the one these students were working with are not the only way to motivate students to program. Whether creating a gaming project, storytelling project, or music video using a programming language, students wrote and created lengthy and complex programs (Adams & Webster, 2013).

In addition to being motivating and interesting to students of young ages, computer programming is an excellent medium in which students can learn and practice important problem solving skills (Cooper, Dann, & Pausch, 2000; Dalton & Goodrum, 1991; Kafai, 1995; Parker, 2012; Suomala, 1996). By giving 3rd, 5th, and 8th graders a series of increasingly more complex tasks, which they had to perform themselves and then “teach” the computer to perform, all students’ proficiency increased and strategies were more easily adapted and adopted to other task problems (Armstrong, 2014). She also noted that student types of questioning evolved throughout the sessions, suggesting self-regulation (i.e., learning to learn) development. Towards the beginning of the study, students asked questions that were directed at the experimenter with no attempt at self-determination, however towards the end students were asking more advanced questions that would guide them to their own answer. Thus, programming can afford opportunities for self-directed learning through observation, imitation, and peer teaching at the point of demand (Baytak & Land, 2011).

Along with helping students’ cognitive ability, computer programming could also benefit students emotionally. Implementing programming into the curriculum can allow for personal expression as students express their ideas, thoughts, and feelings regarding the content they are studying. Baytak and Land (2011) designed a study in which 5th-graders were asked to design video games about environmental problems to teach younger children in their school. The games in which the 5th-graders produced all varied significantly in content and design, however all students developed games that integrated environmental science concepts and finished producing a game in less than 3 weeks that was meaningful to them and the community around them. This suggests that computer programming can be used as a tool for knowledge reformulation and expression for students of all ages and skill levels. Computer game design is a dynamic process where students’ increased experience with game design prompt additional information seeking in both computer knowledge and content knowledge in order to create their games. In programming, students are not placed in different ability groups like typical school learning, enabling them to express themselves as they wish according to level of difficulty, style, or content in their programs. Computer programming allows students of different backgrounds, with different levels of engagement and interest, to come together. This includes gender differences.

Why is it Important for Girls To Learn Computer Science?

Computers and Boys at Early Ages. Games propel boys, at an early age, into the world of computing and reinforce the belief that computing is a male activity. For instance, some of the top games for kids are Minecraft, Rayman Legends, Mario Kart 8, Pokemon Omega Ruby and Alpha Sapphire, and Super Smash Bros. In Minecraft children build structures and battle terrible things, although it does contain some interests for girls such as creating imaginative things by breaking and placing blocks, going on adventures, and
visiting other lands. However, the other top games contain activities such as battling other characters, racing cars, and beating levels, all more boy-stereotyped interests. In addition, their layouts use majority dark or “boy” colors and their characters are more grotesque and gruesome. Expectations from elders for boys to be interested in and good at computing further propel this belief. In addition, computer science curriculum has traditionally reflected boys’ interests and experience levels focusing mainly on violent and realistic games, leaving girls to feel like “outsiders,” wondering how they and what they value can fit into computer culture and curriculum (Margolis & Fisher, 2002). This, along with the common practice of grouping computer science with math and science, both informally and organizationally, may increase the gender gap in computing. Researchers on gender and math have found that self-confidence, not ability, is the significant difference between male and female science students. This lack of self-confidence in the early years could be one of the reasons why girls are less likely than boys to enroll in computer programming classes (Bruckman, Jensen, & DeBonte, 2002). Once enrolled, however, boys and girls have similar levels of programming achievement. In fact, girls performed on average, as well as or better than males, and comprise 60% of the most talented students (Bruckman, Jensen, and DeBonte, 2002; Linn, 1985). Therefore, in order to increase girls’ achievement and thus motivation to enroll in computer programming courses, we need to increase girls’ self-efficacy in computer programming.

Self-Efficacy On Performance and Mastery. As first suggested by the social cognitive theory, in order to increase achievement, educational efforts should focus on raising students’ self-efficacy through authentic mastery experiences (Bandura 1977, 1997). Researchers have since confirmed that self-efficacy plays an influential role in academic performance and mastery (Alfassi, 2003; Cole & Denzine, 2004). It is also believed to be a major contributing factor in the development of students’ attitudes toward subject matter in that students with high self-efficacy regard themselves as being competent and capable and are more apt to be successful (Pajares, 2002). Additionally, Bandura (1977, 1986) drew attention to the connection between self-efficacy beliefs and self-regulatory practices in that the quality of self-regulatory skills students employ depend in part on several underlying beliefs students hold about themselves. Therefore, in order for students to engage in higher self-regulatory skills as programming allows for (Armstrong, 2014; Baytak & Land, 2011), students must have high self-efficacy.

This relates to computer programming in that earlier studies have examined the relationship between self-efficacy and technology acceptance and report that computer self-efficacy increases when technology is utilized frequently as part of the curriculum (DeTure, 2004; Emurian, 2004). However, literature in the area of self-efficacy and technology acceptance at the elementary level is scarce and lacking (Schunk & Pajares, 2004). This is unfortunate in the fact that males and females begin school with a similar interest in technology (Nicholson, Gelpi, Sulzby, & Young, 1998). Yet, somewhere between kindergarten and fifth grade females lose interest in the subject and it is reflected in their computer usage (Butler, 2000). The gender-gap continues to grow with each successive grade level and may affect their choice of college major or career.

Margolis and Fisher (2002) therefore, advocate that a strong sense of self-efficacy is needed in order for young women to be successful in computer science. They state that positive self-efficacy beliefs are directly associated with persistence. If female students judge their computing abilities poorly, they are not likely to persevere in that field. Thus, in
order to increase the likelihood that girls will participate and enroll in computer science classes later on, we need to increase their enjoyment and motivation in computer science by increasing their experience with, and thus their self-efficacy in, computer science.

How Can We Do This?

**Increased Motivation through Storytelling.** A way to increase girls’ participation in computer programming may be to increase their self-confidence in programming. One way to do this is by introducing computer science to girls in a way that is motivating and interesting to them. Margolis and Fisher (2002) determined that an ideal adventure game for girls would feature everyday “real-life” settings as well as new places to explore. They would have a strong story line and leading characters that are everyday people, as real to girls as their best friends. Many studies have confirmed this information, finding that computer programming in the form of storytelling is motivating and interesting for both girls and boys (Kelleher & Pausch, 2006; Kelleher, Pausch, & Kiesler, 2007; Rodger et al., 2009). In the context of a computer camp for girls, Kelleher and Pausch (2006) had middle school girls create stories using Story Alice. They found that all of the girls were able to come up with a story that they were excited about and most girls were enthusiastic about working on their stories. In fact, they found that several girls continued working on their stories during breaks and some even came early on the last day of camp in order to have extra time to finish their stories.

Although this is a promising finding, middle school may be too late to seriously change girls’ attitude about computer programming. Girls in elementary school, generally still like math, but by high school they are less likely than boys to feel competent in math (Margolis & Fisher, 2002). Since computer science is interwoven with math and science, through its use of problem-solving and algorithms, the gender gap between girls and boys in computing could occur during girl’s adolescence (9-15), when girls’ become increasingly aware of the culture that surrounds them. Therefore, by the time girls are being introduced to computer science in middle school, it may be too late to really make a difference in their attitudes and beliefs. A more ideal time to introduce computer science to girls is in the earlier years of elementary school, before girls have really even been introduced to computer science and begun forming their opinions or attitudes. In addition, introducing programming earlier would give girls more experience with computers, therefore putting boys and girls experience with computing on a more even level.

**Increased Experience with Computers.** Bruckman, Jensen, and DeBonte (2002), in a study on high schoolers and college students, found that gender does not affect programming performance. Their study showed that performance is correlated with prior programming experience and time on task. Unfortunately, this puts girls at a disadvantage in comparison to boys. Females who enroll in computer science programs appear to have intellectual skills similar to those of males but less computer experience (Linn, 1985). Boys enter middle school with a great deal of formal and informal computing experience (Bruckman, Jensen, & DeBonte, 2002). By their middle school and high school years, many boys have spent endless hours playing games or manipulating the games so that they do what they want them to do. At all ages from eight up, boys spend at least twice as much time playing computer games than girls (Margolis & Fisher, 2002).
This could be due to the fact that when girls do use technology, they are more focused on the main functionality that they need for their task without further experimentation, where as boys were more explorative and experimental about the different information and communication technology (Hou et al., 2006). Boys are more likely to strive to achieve control over the technology by understanding the construction and functioning, therefore giving them more experience with programming before even entering a computer science class. Girls on the other hand, were more prone to make technology work for their purposes, without understanding analytic details of the technology, giving them less experience with actual programming before computer science is actually introduced into their education.

Introducing computer programming in the early years of elementary school would give girls more experience with programming and computers in general, thus reducing the likelihood of a gender experience gap occurring. Giving girls more experience with computer programming early on could help increase their interest, self-confidence, and motivation to program, thus making it more likely they will enroll in programming classes later on.

Previous Research

As stated in the introduction, there has been successful research on teaching younger students computer science through use of programming languages geared towards children. However, most of the research has focused on students in the upper elementary years and middle school. Very little research looks at teaching computer science in the early elementary years (Fessakis, Gouli, & Mavroudi, 2013; Lee, 2010; Rodger et al., 2009; Suomala, 1996) and even fewer studies actually integrate computer programming into the curriculum (Fessakis, Gouli, & Mavroudi, 2013; Lee, 2010). In addition, most languages such as Alice (Kelleher, Pausch, & Kiesler, 2007) and Scratch (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010) are not intended for children as young as five. Most programming tools target children at least 7-8 years old, but typically older, making them not developmentally appropriate for students in grades K-2. Therefore early childhood education remains an area in which few educational technologies focus on digital creation or high-level thinking. I will now describe the relevant research done in schools and camps with different student-focused programming languages.

**Alice.** Alice is an innovative computer environment specifically designed to introduce students to programming. It allows students to create a three-dimensional world in which designers pose their own stories and problems to solve while also evaluating which solutions are most effective. Students are able to manipulate objects and characters and immediately see results (Cooper, Dann, & Pausch, 2003). Alice was originally developed as an objects-first strategy for teaching computer science courses in high school and colleges (Cooper, Dann, & Pausch, 2003). However, it is incredibly user-friendly, in part because it relies on a drag-and-drop interface to create a virtual world, and has been used successfully on a fifth-grade classroom (Parker, 2012).

Parker (2012), a fifth-grade teacher, integrated Alice in the classroom, where students explored problem solving and math concepts in a real world that they created and controlled. He approached teaching computer programming as a problem-solving unit, in which students would learn the fundamentals of programming, laying the foundation for...
related languages. The unit lasted a week, and he found that by weeks end most students were able to create complex and successful programs. However this was based off of the teacher’s observation and no actual research techniques were employed. There has been no concrete research on the effectiveness of using Alice on students younger than middle school, and still only few that look at students in middle school (Rodger et al., 2009).

However, a storytelling version of Alice has been created, designed for, and successfully used on middle school students (Kelleher & Pausch, 2006; Kelleher, Pausch, & Kiesler, 2007; Rodger et al., 2009; Werner, Denner, & Bliesner, 2009). Storytelling Alice has not only been found to be motivating and interesting to both boys and girls (Kelleher & Pausch, 2006), but it can be used by middle-school students to make games that have the potential to promote aspects of IT fluency (Werner, Denner, & Bliesner, 2009). Research on Storytelling Alice mostly focuses on increasing middle school girls’ motivation for computer science through storytelling, and has not been tried on early elementary students. In addition, studies have not focused on this implementation in the classroom, instead looking at it in the context of either workshops (Kelleher, Pausch, & Kiesler, 2007) or camps (Kelleher & Pausch, 2006). Scratch, on the other hand, has both been studied using elementary students and successfully integrated into language-art projects.

**Scratch.** Scratch is a media-rich programming environment recently developed by the Media Lab at the Massachusetts Institute of Technology (MIT). It is a visual programming environment that allows users, primarily ages 8 to 16, to learn computer programming while working on meaningful projects. Using Scratch, students have created animated stories, games, online news shows, book reports, music video, and more (Maloney et al., 2010). The most prominent feature of Scratch is that it allows even young children to easily create multimedia projects, by putting together visual programmable blocks, similar to jigsaw puzzles, with a click of a mouse button instead of typing complicated programming language constructs on a computer keyboard. Using Scratch, students as young as five were able to learn basic programming concepts (Fessakis, Gouli, & Mavroudi, 2013). However, there is research to suggest that Scratch may still be developmentally inappropriate for younger students, especially under the ages of 8 (Lee, 2010).

Lee (2010) designed a case study to test whether elementary students could learn computer programming concepts and skills, using Scratch, while working on technology enriched language-art projects. She observed a third-grade student using Scratch over a six-month period. She found that while Scratch could help young elementary students learn computer programming concepts and skills while being integrated into the curriculum, there are some concepts and skill that are difficult for young children to grasp. In particular, she found that the student had a hard time implementing event-driven programming and understanding “variable ownership.” She notes that with time and direct instruction, the student was eventually able to create correct event-handling Script without the investigators help and understand variables and their ownership. However, in the classroom the teacher does not have an infinite amount of time to explain these concepts to the students and the students do not have six months to learn computer science. Therefore implementing Scratch into the early elementary classroom may be problematic and Scratch may not be developmentally appropriate for 2nd-graders as a first tool for learning programming. ScratchJr on the other hand, was tailored for early childhood with the
premise that children in kindergarten to second grade can learn and apply programming and problem solving to create interactive animations and stories (Flannery et al., 2013).

**ScratchJr.** ScratchJr allows for student to explore and gain independence in iteratively creating animated and interactive stories. It is a programming language designed to promote early childhood learning outcomes in academic domains, “while also introducing students to computer programming and reinforcing problem-solving and foundational skills” (Flannery et al., 2013). ScratchJr is based off of Scratch with fundamental changes based off of the cognitive, physical, language, and socio-emotional traits of the age range five to seven. In order to make it developmentally appropriate for grades K-2, ScratchJr has “one third fewer programming blocks than in Scratch, only a few other essential tools, and no computer menus to navigate” (Flannery et al., 2013). In addition, the features are laid out intuitively to minimize mouse movement, clicking, and scrolling. Elements of the interface are large enough to facilitate targeting blocks and buttons and labels are primarily icon-based, so early readers can readily learn the elements of the tool.

There has only been one study on the effectiveness of K-2 students learning basic computer programming concepts, as ScratchJr was developed in 2013, but not released to the public until 2014 and is still being upgraded and developed as of 2015 (Flannery et al., 2013). While the authors were creating ScratchJr they consistently tested and refined it based off of their findings of K-2 students using and interacting with Scratch. They then tested the finished product on four classrooms, ranging K-2. They found that all children enthusiastically explored the program and were able to create animated scenes. There were grade level differences in the fact that first and second graders were able to master basic navigation and programming concepts more quickly than kindergartners and had time and cognitive resources to learn and apply more advanced concepts. However, overall ScratchJr has been found to be feasible for both children and teachers to use in the classroom and to support three major categories of learning beyond programming: foundational knowledge structures, discipline-specific knowledge, and problem-solving.

Research Purpose/Questions

Based on the information presented in the previous sections, it would appear that teaching computer programming to early elementary students would be beneficial to all students, especially girls. Focusing on teaching early elementary students provides the opportunity to engage, and more importantly, retain the interest of females in science and engineering as it is typically in the middle-school years that females decide that science and engineering is “not cool” (Rodger et al., 2009). Students who are exposed to computer learning environments have the opportunity to gain higher cognitive skills (Linn, 1985). The fact that girls are poorly represented in courses that have the greatest potential for higher cognitive skill development could end up constraining their career options later on.

Introducing computer science into the early elementary years could increase female exposure to and interest in computer science, perhaps making more motivated, and thus more likely to engage in and enroll in more computer science classes in the future. Exposing girls to computer programming early on and introducing it in a fun and interactive way that aligns with their interests may increase girls’ confidence in programming abilities and thus more motivated to participate in computer programming.
Therefore, my study focused on teaching early elementary students (2nd-graders) computer programming using the developmentally appropriate language called ScratchJr and implementing it into the science curriculum through storytelling. The purpose of my study was twofold. First, my study builds off of Flannery et al. (2013) study, looking at ScratchJr effectiveness in helping 2nd graders understand and comprehend basic computer programming concepts and skills while being integrated into the science curriculum. Second, my study looked at whether there are gender effects in computer science on self-efficacy and attitude, as well as students’ confidence in computers for early elementary students. I examined students’ self-efficacy, attitude, and confidence both before and after the study to see if there was an initial difference and a change in these after teaching students computer programming. I especially focused on girls’ self-efficacy, attitude, and confidence on computers and computer science by introducing computer programming through storytelling. In addition, in order to see if 2nd grade students were capable of learning computer science and mastering basic computer science skills, I administered an assessment both before and after my intervention that tested their knowledge and skills of ScratchJr.

First, I predicted that students as young as 2nd grade would be able to learn and master basic computer science skills in order to create their final project. I also predicted that teaching computer science (CS) using engaging activities (i.e., storytelling) would increase all students’ enjoyment of computer science. As increasing enjoyment was found to increase student’s motivation in computer science (Kelleher & Pausch, 2006; Kelleher, Pausch, & Kiesler, 2007; Rodger et al, 2009), I then predicted that students’ motivation in computer science in general, including girls’ motivation, would increase. Since all students’ exposure to and motivation for computer science would increase, the end result would lead to an increased likelihood that students of both genders would participate and enroll in computer science classes later on. Therefore, my theory of action was:

Thus, my research questions are:
1. Are kids as young as 2nd grade able to master some basic computer science concepts and skills, and use them to serve their own needs?
2. Is there a difference between girls and boys self-efficacy and attitudes towards computers for early elementary students?
3. Does teaching elementary students computer programming increase students self-efficacy and attitude toward computers and computer science?
4. Is there a relationship between students’ academic performance in computer science and their self-efficacy and attitudes in computers and computer science?
Intervention

The population under study consisted of 24 students in a 2nd grade class at Hollywood Elementary School. Out of these students, sixteen are male and eight are female. Two of these students are African American, one is Asian, and the other 21 students are White. The age of these students is between 7 to 8 years old. The majority of students in the classroom were below grade-level, with many students being pulled out for extra help and interventions.

First and foremost, the goal of this intervention was to determine if 2nd grade students were capable of learning and mastering basic computer science concepts and skills. The second goal of this intervention was to increase students’ attitude and self-efficacy, especially girls’, towards computer science, and computers in general. To accomplish this, I taught students computer programming using the developmentally appropriate language ScratchJr, implementing it into the science curriculum through having them create an unhealthy habit message in the form of a story. While the students were learning the basics of computer programming through a series of lessons, they were also learning about and researching health and different unhealthy habits during their science time. The end result of my study was the transformation of their learning and research on unhealthy habits into a written message that was then implemented as a multimedia project using ScratchJr. Their multimedia projects were then put onto a website and shared with the rest of the class as well as the rest of the school, providing students with an idea of how learning computer programming can be beneficial and spread a message much quicker than pencil and paper.

In order to teach the students computer programming, I implemented a series of 9 lessons (see Appendix A), 45 minutes each over the course of 7 weeks, culminating with their interactive message done with a partner. All of the lessons occurred during the last 45 minutes of the day (when they normally had phonics), and did not detract time from the core curriculum. The health lessons and working on their unhealthy message also occurred during this time and during science time in the morning from 10:15 to 10:45 am.

The first 4 weeks consisted of 7 lessons designed to introduce students to ScratchJr, letting them get acquainted and familiar with the program. Each of the lessons built off each other and was designed to introduce the concepts of computer programming slowly and sequentially. The lessons consisted of an assignment or an objective the student must complete in order to master the concepts being taught. The first four lessons were geared toward teaching the students the basics of computer programming (e.g., instructions, sequencing, motion, start and end block, characters, background). The next 3 lessons were focused on concepts that are specifically needed in order to create the students’ interactive messages (i.e., speed, numbers, repeating sequences, speech bubbles, sounds, pages, and wait for). The two weeks were devoted to the researching of their chosen unhealthy habit via the iPad, writing their message, and then making their messages interactive through the creation of their messages using ScratchJr.

Methods

Methodological approach

This study focused on teaching 2nd grade students computer science as well as on increasing girls’ self-efficacy and attitudes towards computer science, in the hopes that...
Computer Science and Self-Efficacy

doing so could increase girls’ future enrollment and representation in computer science in the upper grades and computer science job field. Therefore, since my research is a specific educational problem that needs to be solved, I used action research in order to determine the successfulness of my study. In conducting this study I hoped to emphasize the importance of teaching computer science to early elementary students, girls in particular, and improve the practices of teachers and administrators on teaching computer science to early elementary students. More specifically, my study is based on practical action research, as it seeks to research the effect of teaching computer programming on students’ self-efficacy and attitudes. These results will then inform teachers and administrators on the effectiveness of teaching computer programming in increasing girls’ self-efficacy and attitude on computers and computer science.

In my action research, I collected quantitative data, making it a quantitative research design. I focused on quantitative data, using informal questions to individual students throughout the study in order confirm the results of the survey and pre-post tests and further my understanding of students experience with and engagement in learning computer programming. The answers to these questions will not be used in my analysis but as support for my findings.

Data collection
This study focused on the success of teaching 2nd grade students computer science. It also focused on teaching young elementary students computer science in order to increase students’, especially girls’, self-efficacy and attitudes towards computer science and computers in general. In order to determine if my intervention was successful, students took a 21-item survey of their attitudes and self-efficacy towards computers and computer science (see Appendix B). Of those 21 items, 6 items were aimed at determining students’ self-efficacy in computer science, 5-items were on students’ attitudes towards computer science, 5 items looked at students’ self-efficacy in computers, and the last 5 items were used to determine students’ attitudes towards computers. Each item was assessed on a scale ranging from 1 (Strongly Agree) to 5 (Strongly Disagree). On the assessment, lower average scores indicated higher levels of self-efficacy and attitudes towards computers or computer science.

The survey was given twice, once before the intervention and once after the intervention. I began my study with the survey in order to determine students existing attitudes and self-efficacy towards computers and computer science. Questions on the before survey and after survey were aimed at research question 2, and the results of the after survey were aimed at research question 3.

In addition, a pre-test on basic computer programming concepts through the ScratchJr environment (see Appendix C), in order to determine students’ success and understanding of the language was necessary (research question 1). This was also used in order to determine if teaching computer science to young elementary students affects students’ understanding of the language, which in turn affects students’ self-efficacy and attitudes. Questions on the post-test along with their post-survey answers were aimed at research question 4.

Throughout my intervention I conducted individual, brief, informal, undocumented interviews with students to gather more data on their experience with and interest in computers and computer science. These questions were asked during their programming
time. I primarily asked questions pertaining to research questions 2 and 3. As these interviews were undocumented I used them to confirm or explain my findings, however they are not recorded as part of my research. After compiling and evaluating the results of the survey and pretest, I conducted my intervention. Throughout my intervention I took notes on observations as well as informally interviewed the students on their experience, enjoyment, and thoughts about ScratchJr and learning computer programming during their programming time. These notes and observations were for my own knowledge and to help me inform my teaching of ScratchJr in the future.

After my intervention, I again gave students the survey and a post-test. The after survey was used to see if there was any difference between students’ self-efficacy and attitudes on computers and computer science before learning computer programming and after, as well as to see if there was a difference between gender on self-efficacy and attitudes. The post-test was used along with the pre-test to determine how effective teaching students computer science was and whether or not there was a relationship between students’ understanding of computer science after learning it and their self-efficacy and attitudes towards computers and computer science. Finally, my informal observations and interviews were used to confirm the information gathered from the surveys and pre-post tests and triangulate the data collected.

Table 1: Research Questions and Data Sources

<table>
<thead>
<tr>
<th>Research Questions</th>
<th>Data Source 1</th>
<th>Data Source 2</th>
<th>Data Source 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are kids as young as 2nd grade able to master some basic computer science concepts and skills, and use them to serve their own needs?</td>
<td>Pre-Post ScratchJr Test</td>
<td>Observations</td>
<td>Final Project (Interactive ScratchJr Unhealthy Habit Message)</td>
</tr>
<tr>
<td>Is there a difference between girls and boys self-efficacy and attitude towards computers for early elementary students?</td>
<td>Pre-Post Self-Efficacy and Attitude Survey</td>
<td>Field Notes</td>
<td></td>
</tr>
<tr>
<td>Does teaching elementary students computer programming increase students self-efficacy and attitude towards computers and</td>
<td>Pre-Post Self-Efficacy and Attitude Survey</td>
<td>Field Notes</td>
<td></td>
</tr>
</tbody>
</table>
## Computer Science and Self-Efficacy

<table>
<thead>
<tr>
<th>computer science?</th>
<th>Pre-Post ScratchJr Test</th>
<th>Pre-Post Self-Efficacy and Attitude Survey</th>
<th>Field Notes</th>
</tr>
</thead>
</table>

**Data analysis**

This study employed a quantitative research design. Quantitative data was analyzed using t-tests. Informal observations and interviews were used to triangulate the quantitative data. I conducted a one-tailed, paired samples t-test using their pre- and post-ScratchJr assessments in order to determine if students were successful in learning and mastering some basic computer science concepts and skills (Question 1). Using the before survey I administered a two-tailed, independent samples t-test to determine whether there are initial gender differences on self-efficacy and attitudes on computers and computer science, as well as performance on computer programming (Question 2). Then in order to determine if there were gender differences after learning computer programming, administered a two-tailed, independent-samples t-test using the survey given after the intervention (Question 2). To determine the effect teaching computer programming has on students’ self-efficacy and attitudes, two-tailed, paired-sample t-tests were administered (Question 3). Finally, in order to determine if students’ understanding and performance in computer programming is related to their self-efficacy and attitudes on computers and computer science after learning computer programming (Question 4), I administered a Pearson product correlation. I reviewed my observations and informal interviews for evidence that supports or contradicts my quantitative findings.

**Validity concerns**

In order to triangulate and further increase the validity of my data collected from the surveys and pre-post tests, I observed the students while they were working and informally recorded my observations, as well as informally interviewed some of the students on their experiences and thoughts about computers and computer science. In addition, to account for my own personal biases, I consulted and debriefed with my mentor teacher on her thoughts about the students’ interests, progress, and performance in computer programming. However my notes could still be skewed due to sample selection, as I was unable to ask all students and I selected the ones I questioned. To account for this, my mentor teacher tallied the students I talked to most so that I could see which students’ opinions I was not getting.

To ensure the content validity of my pre-post tests, I used the assessments that came with ScratchJr. These assessments were made by the creators and were designed to
test students’ knowledge of computer programming based on the ScratchJr language. In this way, I can be sure that they test students’ knowledge in a fair and understandable way; based on the language the students’ were learning.

To ensure internal validity of the attitude survey questions on computers and computer science, I pulled and adapted questions from the Children’s Attitude to Computer Questionnaire (Todman & Dick, 1993) that has been tested for internal validity. To ensure internal validity of the self-efficacy survey questions, I consulted Bandura’s “Guide for Constructing Self-Efficacy Scales” and agreed upon a series of questions with my professor who is a researcher in self-efficacy, David Morris. However, since there has yet to be a study that examines students’ self-efficacy and attitude in computers and computer programming, I had to pick and choose, as well as create my own survey questions that got at my research questions in order to create my survey. Thus, the survey I created could have less validity. However, I consulted with two Master’s in the Art of Teaching (MAT) professors on my survey questions and whether or not they appeared to have face validity. Therefore, I am quite confident that my survey fits my research questions/hypotheses.

Findings and interpretations

In this section, I have organized the analyses to answer each of my research questions. Quantitative results are reported along with answers to my informal questions when available, which allowed me to triangulate the data and support my results. Following the statistical results for each research question are my interpretations of the meaning of these findings, which is grounded in relevant research.

Which Students Were Included in Analyses?

Although all of the 24 students participated in the exploration and learning of computer programming using ScratchJr throughout the course of the unit and completed the culminating project, two students were removed from analyses due to absences and a resulting lack of data. Complete data existed for 22 students, 15 of whom were male.

Are kids as young as 2nd grade able to master some basic computer science concepts and skills, and use them to serve their own needs?

Quantitative analysis. To date there is only one study that has shown students as young as kindergarten and first-grade are capable of learning basic computer programming concepts and skills through the use of ScratchJr (Flannery et al., 2013). Therefore, my first goal was to confirm these results and extend this finding to include 2nd grade students. In order to determine if students as young as 7 or 8 were able to learn basic computer programming skills and concepts through ScratchJr, I looked at students’ academic performance on the pre- and post-ScratchJr assessments. I first calculated students’ overall average performance on the test before and after the intervention (see Appendix C). Then, I conducted a one-tailed paired sample t-test to determine if there was a significant difference in students’ academic performance before and after learning ScratchJr. A one-tailed t-test was used because I predicted that students’ scores would improve on the ScratchJr assessment after my intervention as opposed to before. Additionally, as I was looking at students’ scores before and after the intervention, a paired samples t-test allowed me to look at these same samples both before and after the intervention in order to
compare students’ scores. The alpha level was set at .05, which indicates that there is less than a 5% chance that the change in score was due to chance or random error.

The t-test revealed a significant difference in students’ scores on the ScratchJr test before and after the intervention ($p < .05$). There was a significant increase in students’ scores after the intervention ($M = 76\%$) compared with their scores before the intervention ($M = 20\%$). After learning computer science using ScratchJr, students performed better on the ScratchJr assessment than they had before learning computer science (see Figure 1).

**Figure 1:** Students’ academic performance on the ScratchJr assessment before and after intervention.

**Observations.** My observations throughout the unit support this finding. Towards the beginning of the unit, students were hesitant to explore ScratchJr and I spent the majority of their free-time exploration going around and answering questions, clarifying what I had taught, and re-teaching students the concepts they had just learned. However, towards the end of the unit I spent much more time going around and looking at students’ projects. As the unit progressed students got more and more excited to show me what they had created and the questions on clarity and programming concepts decreased. In addition, towards the end of the unit, when students would ask me how to do something, instead of outright giving them the answer I was able to scaffold them in a way in which they figured the answer out on their own by asking leading questions. Students became much more independent and confident in their skills as the unit progressed and I saw a marked improvement in their programs as a result.

**Final project.** The unit ended with students creating, with a partner, an interactive, multimedia message on an unhealthy habit of their choosing using ScratchJr. All groups succeeded in creating a four-panel ScratchJr project in which their characters moved and interacted with each other. Not only did their characters move and interact, but also they did so in a purposeful and meaningful way that went along with the message the children wrote and then recorded into their ScratchJr projects. Students mastered basic computer
programming concepts and then used them to build their stories about unhealthy habits (their projects can be found at: http://healthyhabitsscratchjr.blogspot.com/ or using the QR code below). The majority of students were able to master and then use basic computer programming concepts and skills in their projects such as movement, changing size, fading in and out, playing a recorded sound, word output, and simple looping (repetition).

**Interpretation.** These findings support the Flannery et al. (2013) finding that students as young as early elementary school are capable of learning and mastering basic computer programming concepts and skills. Students as young as 7 and 8 can successfully be taught computer science. My second grade students were not only able to learn basic computer programming concepts, but also they were able to master them to the point that they could use them in order to make their characters move and interact in a way that was meaningful and purposeful for their unhealthy habit message.

Is there a difference between girls and boys self-efficacy and attitudes towards computers for early elementary students?

My second goal was twofold in that I wanted to see if there was an initial difference in students' self-efficacy and attitudes towards computers before my intervention as well as if there was a difference after having been taught computer science (after my intervention).

**Before intervention.** Earlier research, although quite dated, still seems to hold true today based on my results. This research indicated that males and females begin school with a similar interest in technology; yet somewhere between kindergarten and fifth grade females lost interest in the subject (Butler, 2000). Therefore, in order to help figure out whether young girls have less interest in technology, I looked at 2nd-grade girls and boys' initial attitudes and self-efficacy in computers. I first calculated students' overall self-efficacy using their average score on the 5 items that measured computer self-efficacy in the 21-item questionnaire (see Appendix B). Then, I conducted a two-tailed independent samples t-test to determine if there was a significant difference between girls and boys' self-reported self-efficacy before I taught them computer science. A two-tailed t-test was used because I could not predict if there was a difference between boys' and girls' self-efficacy and in what direction it might be (i.e., boys have higher self-efficacy than girls or girls have higher self-efficacy than boys). Additionally, as I was looking at two different samples (girls and boys), an independent samples t-test allowed me to look at these separate samples in order to compare students' scores.

The t-test revealed no significant difference in self-reported self-efficacy between girls ($M = 9.57, SD = 2.44$) and boys ($M = 8.06, SD = 1.88$). At the start of the study, before learning computer science, both boys and girls had similar self-efficacy towards computers (see Table 2).

Next, I calculated students' overall attitudes towards computers using their average score on the 5-items that measured attitude in the 21-item questionnaire (see Appendix B).
I then proceeded to repeat the tests done above for self-efficacy, this time using their average attitude scores. The t-test revealed no significant difference in self-reported attitude between girls ($M = 9.29, SD = 4.54$) and boys ($M = 7.31, SD = 2.21$). At the start of the study, before learning computer science, both boys and girls had similar attitude towards computers (see Table 2).

Table 2:

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Boys</th>
<th>Girls</th>
<th>Mean</th>
<th>SD</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Efficacy</td>
<td>16</td>
<td>7</td>
<td>9.57</td>
<td>2.44</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td>8.06</td>
<td>1.88</td>
<td></td>
</tr>
<tr>
<td>Attitude</td>
<td>16</td>
<td>7</td>
<td>9.29</td>
<td>4.54</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td>7.31</td>
<td>2.21</td>
<td></td>
</tr>
</tbody>
</table>

**Interpretation.** These findings support that females and males begin school with a similar interest in technology (Nicholson et al., 1998) and while maintaining the possibility that somewhere between kindergarten and fifth grade females lose interest in the subject (Butler, 2000). As the findings were not significant it means that girls’ and boys’ self-efficacy and attitude, while different, were not different enough to be considered important. While at the same time, although not significant, compared to the boys in my 2nd-grade class, girls did tend to have lower self-efficacy and attitude towards computers. This is strengthened further by the informal questions I asked at the onset of the study and the comments heard before beginning. Before even beginning my study I could see there was a big difference in the use of computers and technology between girls and boys. Many of the boys in my class constantly talked about Minecraft, a popular video game, and read books about Minecraft during Read-to-Yourself time during language arts rotations. While some girls did show an interest in Minecraft, it was not to the same extent as the boys. In addition, when asked about the types of activities students were planning on doing at home, many more of the boys responded with video games or Minecraft, where as the girls were more varied in their responses. Again it must be noted that the small sample of girls in this sample size could be a factor in their varied responses. As Minecraft is a video game that can be played on the computer, it supports Butler’s (2000) finding that males tend to be more interested in computers and technology then females. In addition, it supports the Bruckman, Jensen, and DeBonte (2002) finding that males have more experience with and exposure to technology than females, which is found to be related to students’ programming performance.

Overall these findings support that boys tend to have more exposure to and experience to video games and computers compared with girls, even at this early age (Margolis & Fisher, 2002).
After intervention. DeTure (2004) and Emurian (2004) found that self-efficacy increases when technology is frequently used as part of the curriculum. Therefore, I looked at girls and boys self-efficacy and attitudes towards computers after having been taught computer programming through ScratchJr over the course of a couple of weeks. I first calculated students’ overall self-efficacy using their average score on the 5-items that measured computer self-efficacy in the 21-item questionnaire (see Appendix B). Then, I conducted a two-tailed independent samples t-test to determine if there was a significant difference between girls and boys’ self-reported self-efficacy after having taught them computer science.

The t-test revealed no significant difference in self-reported self-efficacy between girls’ \((M = 9.57, SD = 4.31)\) and boys \((M = 9.25, SD = 2.57)\). After having been taught computer science and exposed to technology in the classroom regularly, both boys and girls had similar self-efficacy towards computers (see Table 3).

Next, I calculated students’ overall attitudes towards computers after being taught ScratchJr using their average score on the 5-items that measured attitude in the 21-item questionnaire (see Appendix B). I again conducted a two-tailed independent samples t-test to determine if there was a significant difference between girls and boys self-reported attitudes towards computers after having been taught computer science.

The t-test revealed no significant difference in self-reported attitude between girls \((M = 7.86, SD = 3.08)\) and boys \((M = 7.81, SD = 2.71)\). After having been taught computer science and exposed to technology regularly as part of the curriculum, both boys and girls had similar attitude towards computers (see Table 3).

<p>| Table 3: Mean Scores For Self-Efficacy and Attitude Scores on the Post-Survey |
|-----------------------------------------------|--------|--------|--------|--------|</p>
<table>
<thead>
<tr>
<th>Measurement</th>
<th>n</th>
<th>Mean</th>
<th>SD</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Efficacy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boys</td>
<td>16</td>
<td>9.57</td>
<td>4.31</td>
<td>0.86</td>
</tr>
<tr>
<td>Girls</td>
<td>7</td>
<td>9.25</td>
<td>2.57</td>
<td></td>
</tr>
<tr>
<td>Attitude</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boys</td>
<td>16</td>
<td>7.86</td>
<td>3.08</td>
<td>0.31</td>
</tr>
<tr>
<td>Girls</td>
<td>7</td>
<td>7.81</td>
<td>2.71</td>
<td></td>
</tr>
</tbody>
</table>

Interpretation. These results indicate that after learning computer science boys and girls had relatively the same self-efficacy and attitude towards computers. This is slightly different than at the onset of my study. While it was not significant, at the beginning of my study girls’ had lower self-efficacy and attitude towards computers than boys. Therefore, this result is quite promising in the fact that there was no difference between gender on self-efficacy and attitude indicating that teaching early elementary school students computer science could help decrease the gap that appears to occur between kindergarten and fifth grade where females lose interest in computers (Butler, 2000).
If the patterns I detected—girls with much lower scores before the intervention; girls’ and boys’ scores matching after—were to hold true with larger sample sizes, it would indicate that teaching computer science in early elementary school could help decrease the gap in interest in technology and computers that occurs between boys and girls between kindergarten and fifth grade.

Does teaching elementary students computer programming increase students self-efficacy and attitude toward computers and computer science?

**Quantitative Analysis.** As self-efficacy and attitude plays an influential role in academic performance and mastery (Alfassi, 2003; Cole & Denzine, 2004; Pajares, 2002), the main purpose of my study was to increase students’ self-efficacy and attitude towards computer science and computers, especially those of girls, through teaching them computer programming in the hopes that they will be more likely to pursue and persistence in the subject later on in life.

In order to determine whether teaching computer science through ScratchJr affected students’ self-efficacy and attitude towards computer science and computers, I conducted four tests. To determine students’ self-efficacy towards computers science, attitude toward computer science, self-efficacy towards computers, and attitudes towards computers, I summed the items on the questionnaire pertaining to each component (see Appendix B for coding details). Then, I conducted four two-tailed t-tests, much as was described in the previous analysis, to determine if there were significant differences between the pre-test and post-test scores (see Table 4). However, in this instance I used a paired samples t-test instead of an independent samples t-test. The pre-test and post-test questionnaire were identical, allowing for a paired samples t-test to directly compare students’ scores.

The results of the t-tests indicate that there were significant differences between the pre-test and post-test for students’ self-efficacy and attitude towards computer science (both *p*-values < .05). This means that the implementation of teaching students computer science through ScratchJr increased students’ self-efficacy and attitude in computer science. These results support my theory of action and reflect that teaching computer science through engaging activities increases students’ enjoyment and motivation for computer science.

However, the results of the t-test also indicate that there were no significant differences between pre-test and post-test for students’ self-efficacy and attitude towards computers (both *p*-values > .05). This means that the implementation of teaching students computer science through ScratchJr did not change students’ self-efficacy nor attitude towards computers.

Table 4:

<table>
<thead>
<tr>
<th>Component</th>
<th>Pretest</th>
<th>Posttest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science Self-Efficacy</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Mean</td>
<td>16.57</td>
<td>10.93</td>
</tr>
<tr>
<td>SD</td>
<td>1.65</td>
<td>3.29</td>
</tr>
<tr>
<td><em>p</em></td>
<td>&lt; .05</td>
<td></td>
</tr>
</tbody>
</table>

**Mean Scores For Each Component of Self-Efficacy and Attitude on the Pre- and Post-test**
Interpretations. This result confirms the initial ideas behind my intervention in which teaching students computer science in an engaging and meaningful way increases students’ engagement and motivation for computer science through increasing students’ self-efficacy and attitudes towards computer science. It was further strengthened in regards to increasing girls engagement in computer science as two out of the eight girls in my class continuously came up to me throughout the unit and told me how they had persuaded their parents to get ScratchJr on the iPad at home and how they were creating projects at home. However in order to determine if increasing students' self-efficacy and attitudes towards computers science increases their likelihood of participating and enrolling in computer science classes later on I would have had to employ a longitudinal study in which I continued to examine students interest in and participation in computer science.

These results also show that teaching computer science did not affect students’ self-efficacy and attitudes towards computers. Students maintained approximately the same self-efficacy and attitude towards computers after learning computer science as they had before. This could be due to the fact that at the start of the study, students’ self-efficacy ($M = 8.52$) and attitude ($M = 7.91$) towards computers were already very high. Simsek (2011) found that elementary students had higher self-efficacy scores towards computers then secondary, therefore a ceiling effect could have occurred here, where the students’ self-efficacy and attitude towards computers was already very high.

Is there a relationship between students’ academic performance in computer science and their self-efficacy and attitudes in computers and computer science?

Quantitative Analysis. Finally, I thought that how well students understood ScratchJr could affect their self-efficacy and attitude towards computers and computer science. In order to determine if this was a factor in students’ self-efficacy and attitude towards computers and computer science after having learned it, I first summed students scores on each of the items on the post-questionnaire that pertained to the four different components (i.e., computer science self-efficacy, computer science attitude, computer self-efficacy, computer attitude) and calculated students’ grade on the ScratchJr post-test (see Appendix C). Then, I conducted a Pearson correlation to determine if there was a relationship between students grades on the ScratchJr test and their final self-efficacy and
attitude towards computers and computer science. As these are both quantitative variables, a Pearson correlation was used in order to determine if there was a relationship between these two variables and if there was, the strength of the relationship.

The results indicated that there was a weak positive relationship between students’ grades on the ScratchJr test and their self-efficacy towards computers, \( r = .26 \). Students who did better on the ScratchJr test tended to have a higher sense of self-efficacy towards computers (see Figure 2).

The results also indicate that there was no relationship between students’ grades on the ScratchJr test and their self-efficacy and attitude towards computer science as well as their attitude towards computers. Students’ grades on the ScratchJr test did not seem to relate to their attitudes towards computer science and computers and their self-efficacy towards computer science.

![Figure 2: The relationship between students’ grades on the ScratchJr test and their scores on the computer self-efficacy component of the post-survey.](image)

**Interpretations.** As self-efficacy plays an influential role in academic performance and mastery (Alfassi, 2003; Cole & Denzine, 2004), I wanted to see if students’ self-efficacy and attitudes were related to their academic performance on the ScratchJr test. I thought that their mastery of ScratchJr might help to explain their self-efficacy and attitude towards computers and computer science. In the case of students’ self-efficacy towards computers, there does seem to be some relationship between their self-efficacy and their academic performance. Students who performed well on the ScratchJr test tended to have higher self-efficacy towards computers. However, academic performance had no relationship with their attitude towards computers and computer science, as well as no relationship with their self-efficacy in computer science.

**Discussion of results**

My results indicate that teaching early elementary students (2nd graders) computer programming through the use of the developmentally appropriate language ScratchJr and
implementing it into the science curriculum through storytelling was effective. My results confirm and expand off of the Flannery et al. (2013) study, showing that students were able to learn and understand ScratchJr, as evidenced through their academic performance on the before and after ScratchJr tests. Teaching students computer science not only exposed them to the subject, but increased their self-efficacy and attitude towards computer science in general. At the beginning of my study, when asked who had heard the term computer science and computer programming, not one student raised their hand. By the end of my study many of the students would come up to me after specials each day asking if we were going to computer program this afternoon.

In addition, my results confirmed that although not significant given my sample size, there may be gender differences in students’ self-efficacy and attitude towards computers at the onset of my study showing that by 2nd-grade, girls are already starting to lose interest in computers compared to boys (Butler, 2000). Girls’ tendency to have lower self-efficacy and attitude towards computer science than boys before learning computer science. However, after learning computer science through ScratchJr, girls’ and boys’ had similar attitudes and self-efficacy towards computer science. This result is promising in itself as, if it holds true in a larger sample, it could mean that teaching computer science to early elementary school students could help decrease the gap that occurs between boys and girls interest in technology from kindergarten to fifth grade.

Although the increase was not significant, the girls’ attitudes towards computers increased after learning ScratchJr. However, what was really surprising was that boys tended to decrease in self-efficacy towards computers after having learned ScratchJr. This could be due to the fact that elementary students were found to have lower computer-anxiety scores and higher computer self-efficacy scores than secondary students (Simsek, 2011). As of elementary school, students’ interactions with computers have been relatively simplistic. They use them for games or occasionally for work, but they have yet to delve deeper into the mechanics and working of computers. As boys typically have more experience and exposure to computers (Margolis & Fisher, 2002), they may believe themselves to be more of an expert on the computer than girls. However, after learning computer science, the boys may have realized they are not as experienced with computers as they initially believed. Therefore, teaching them computer science may have given the boys in my class a more realistic view of their competence in computer science, which would explain why their self-efficacy towards computers went decreased after learning ScratchJr. They may have found out that working with computers are harder than they originally thought.

Overall, my results indicate that my unit was good for the present as my students were able to create interactive, multimedia unhealthy habit messages in order to spread the knowledge of unhealthy habits to the rest of the student body (i.e., my students were able to do something they wanted to do using new-found skills). In addition, it is likely good for the future as well since it makes sense that improved self-efficacy could lead to an increased likelihood of pursuing computer science in the future, and eliminating gender differences means this could be true for the boys and girls in my class.

Conclusion

The purpose of this study was confirm that students in early elementary can learn computer science, as well as to introduce and expose students, especially girls, to computer
science in the early elementary years. Based on the literature I reviewed, I found that
teaching computer science to early elementary school students through engaging activities
and using a developmentally appropriate language could be beneficial in increasing
students’ exposure and self-efficacy towards computer science. This could increase the
likelihood of them enrolling in or engaging in computer science in the future. From the data
I collected during the short implementation of teaching them Scratch Jr., I can conclude that
teaching my students’ computer programming increased their attitude and self-efficacy
towards computer science. In light of these findings, I will continue to integrate computer
programming and computer science into my teaching to help promote and expose early
elementary students to computer science in the hopes that they will be more likely to
enroll in computer programming courses in the future.

Limitations

Although I can draw these conclusions for this particular group of students, these
findings cannot be generalized to other populations. I was limited by a small sample size of
24 students, only 8 of whom were girls, with the extraction of 1 girl due to lack of data,
making for a total of only 7 girls. Repeating this study with a larger sample size would
increase the reliability and validity of the results. Additionally, repeating this study with
other grade levels would provide interesting information for comparison, especially trying
it with even younger grade levels such as kindergarten or 1st grade.

Another limitation was the length of my study. Although I implemented my study
over the course of 7 weeks, the lessons were sporadic and for short periods of time (45
minutes or less), making it very difficult to both teach them new information and allow
time for exploration and creation of programs using the new information I taught them. I
tried my best to get in at least 3 lessons each week of actual computer programming
through Scratch Jr. However, some weeks this was not possible due to uncontrollable
factors, making the unit last longer than the 4 weeks I had originally planned.

Furthermore, the amount of time for each lesson was a problem. The lessons were
originally planned as 9 one-hour lessons. However, this was not possible in the current
setting as the only opening in the students’ schedule to implement my study was during the
last 45 minutes of the day. In order to scale the lessons down into the 45 minute time
frames, which ended up being 30 minutes after technology transition time, I ended up
having to gloss over a lot of the lessons in order to allow them time to explore and
experiment. Ideally, I would have liked to have 9 lessons each around an hour or longer so
that I could thoroughly explain the new information I taught them and have them practice
making programs using this new information. However, it ended up being around 9 lessons
where the instruction lasted half an hour or less. Even though the students are on the
higher end of the age spectrum for Scratch Jr, which is developmentally appropriate for K-
2, and may be able to grasp the concepts quicker, they still needed more time to experiment
and practice the new concepts I had taught at the beginning of class.

Finally, as there has never been a study on early elementary students self-efficacy
and attitudes towards computer programming, the questionnaire I created was pieced
together from multiple other questionnaires and may have been too complex for second
grade students. The wording, particularly on the reverse-coded items, confused many
students and may have affected their ability to accurately report their engagement.
Creation of a simplified self-efficacy and attitude survey would allow for younger students to participate in this research.

Recommendations

For future research. To extend the results of this study, future research should examine whether introducing computer science in the early elementary years increases students’ likelihood of enrolling in and engaging in computer science courses in the future through the use of a longitudinal study. This research would be particularly beneficial for increasing the enrollment in computer science courses as well as increasing the number of computer science majors. Additionally, if possible, future research should look at the best ways for introducing computer science to girls and whether or not exposing them to computer science earlier increases their confidence in computers and technology and makes them more likely to believe that computer science is for both boys and girls. Finally, if possible, future research should be experimental. This would allow for comparisons between two groups learning the same content through different methods. Through using this methodology, the researcher could conduct independent samples t-tests to compare students’ self-efficacy and attitudes towards computers and computer science between groups.

For teachers. I am unable to generalize the results of this study; however, I believe that teaching computer science to early elementary schools has promise in all classrooms and in all grade levels. Whereas my study was conducted in a second grade classroom with very few females, there is literature to support the effectiveness of teaching computer science in kindergarten (Flannery et al., 2013), first-grade (Flannery et al., 2013; Nicholson, Gelpi, Sulzby, & Young, 1998), and even fifth grade classrooms (Parker, 2012). This literature helps support teaching computer science as its own subject as an effective tool across grade levels for both genders. I would suggest infusing and integrating the learning of computer science into all subjects, as it can be used and is applicable in all areas of life.

Implications

I believe that the information gained from this study is of great value to the educational field. During the seven weeks in which I taught students computer programming through ScratchJr, the majority of my students were excited for the end of the day and engaged in all of the lessons as well as during free time in which they were able to explore the application on their own. They were having fun and were excited to start ScratchJr each day. Overall, they followed behavioral expectations when using the iPads, collaborated when working on their messages with their partners, and persevered through the creation of their interactive messages. Instead of counting down the minutes to dismissal, they wanted to keep “playing.” Not only were they engaged during ScratchJr, but also even during the tedious task of writing their messages to make into an interactive video, students worked hard to create a writing and message they would be proud to share. They also achieved significant growth on the ScratchJr assessment, which supports that they learned too. For me, this was the best way to end the day: happy, engaged students who are excited to learn.
References


DeTure, Monica. (2004). Cognitive Style and Self-Efficacy: Predicting Student Success in Online Distance Education. American Journal of Distance Education, 18(1), 21-38.


Scratch Lesson 3

Scratch Lesson 4

Scratch Lesson 5

RISING TIDE VOLUME 8
## Appendix B

### Self-Efficacy and Attitudes on Computers and Computer Science Survey

#### Computer Science Self-Efficacy

**I am confident I could use ScratchJr to make stories on my own.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**I am confident I could use ScratchJr to make games or other types of programs.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**I am confident using ScratchJr to make my own stories**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**When I do something new on ScratchJr, I am confident that I can do well.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**I am not confident in using ScratchJr.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**I am confident I could make the characters in ScratchJr do what I want them to do.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

#### Computer Science Attitude

**Writing my own computer games and stories using ScratchJr is fun. (fun)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**I would use ScratchJr at home on my own. (fun)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**It is difficult to learn ScratchJr. (ease of use)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**I don’t want to learn more about ScratchJr. (usefulness)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**ScratchJr is stupid. (usefulness)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

#### Computer Self-Efficacy

**I can use computers to do what I want to do.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

**When I am unsure of how to do something on the computer, I am confident I can figure it out.**

---

RISING TIDE VOLUME 8
### Computer Science and Self-Efficacy

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

I can use computers to do things other than playing games or writing things for class.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

When I do something new on the computer, I am confident that I can do well.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

I am not confident using computers.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

**Computer Attitude**

**Computers make learning fun. (fun)**

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

**Computers are stupid. (usefulness)**

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

**I like using computers after school. (fun)**

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

**I don’t want to learn more about computers. (usefulness)**

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

**It is difficult to use a computer. (ease of use)**

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
</tr>
</tbody>
</table>

**Interview Questions**

(Note: I will not ask each student I interview all of these questions, just a couple to get a general sense)

1. Have you ever heard about computer programming? Do you know what it is?
2. What did you make the characters in ScratchJr do?
3. Was it easy or hard to make these characters in ScratchJr do ________?
4. What did you learn about making the characters in ScratchJr do ________?
5. Did it get easier to make the characters move as you kept playing with ScratchJr?
6. Would you play with ScratchJr outside of school? What would you try to do?
7. Is there anything else you would like to learn how to do in ScratchJr?
Appendix C

Pre-Post Test for ScratchJr

**1. Overview**
This assessment aims to assess students’ understanding of the programming blocks in the ScratchJr environment. The assessment was designed to evaluate their knowledge and skills by presenting them with various programming challenges.

**2. Setting Up**
To prepare for this assessment, students need to:
- Familiarize themselves with the ScratchJr interface.
- Understand the basic blocks available in ScratchJr.
- Practice sequencing and branching concepts.

**3. Assessment:**
Students use the ScratchJr app to complete the following tasks:
- Create a program that moves a character to a specific location.
- Use the "pen" tool to create a drawing.
- Add sound effects to their programs.

**4. Moving the Assessment**
Some tips for moving the assessment:
- Encourage students to think critically about the blocks they use.
- Guide them to explore different combinations of blocks to achieve the desired outcome.

---

**Name:**

---

**1. **
- **2. **
- **3. **
- **4. **
- **5. **
- **6. **
- **7. **