

Cats in Space, Pigs that Race: Does self-regulation play a role when kindergartners learn to code?

A dissertation

submitted by

Elizabeth R. Kazakoff

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Child Development

TUFTS UNIVERSITY

May 2014

© 2014 Elizabeth R. Kazakoff

Dissertation Committee:

Marina Umaschi Bers, Ph.D., Tufts University (Chair)

Richard M. Lerner, Ph.D., Tufts University

Megan McClelland, Ph.D., Oregon State University

Karen Brennan, Ph.D., Harvard University

Abstract

This dissertation examined the bidirectional relationship between self-regulation and learning to code in two kindergarten classrooms utilizing ScratchJr, a developmentally appropriate programming language software tool for children ages five to seven. This pilot study included 38 students, all of who participated in an eight-lesson ScratchJr curriculum. The program measured students' skills and success in three ways: Programming Score, Goal Completion Score, and Time on Task Score. The coding skill variables were compared to the participants' initial level of self-regulation as measured by the Head-Toes-Knees-Shoulders (HTKS) assessment. Results indicated that overall children in this study attempted and understood ScratchJr programming blocks and interface elements, regardless of initial self-regulation level ($M = 2.95$, $SD = .79$; $N = 35$, $r(33) = .14$, $p < .41$, 95% CI = [-.20, .46]). However, there were some differences in Goal Completion Score compared to initial level of self-regulation. There was a moderate correlation when comparing the average Goal Completion Score to initial level of self-regulation as measured by the HTKS pre-test ($n = 34$, $r(32) = .36$, $p = .03$, 95% CI [.03, .63]). This study also examined whether learning to code could impact self-regulation skills. In order to examine this question, one classroom participated in eight lessons of ScratchJr ($n = 20$), and a second classroom participated in sixteen lessons ($n = 18$). Both groups were pre-tested and post-tested on self-regulation skills before and after completion of the ScratchJr curriculum. Results indicated there might be a relationship between longer exposure to practicing coding in ScratchJr and an increase in self-

regulation, as the 16-lesson group had an average self-regulation score increase of 23.0% versus the 8-lesson group whose self-regulation scores increased by 4.9%. However, as these results were not significant ($F(1, 33) = 2.23, p = .14, n^2_{\text{partial}} = .06, 95\% \text{ CI } [-1.2, 8.0]$), they should be interpreted with caution. Limitations of the current study, future directions for the study of programming languages, understanding how children learn to code, and implications for curriculum development were addressed.

Keywords: coding, programming, self-regulation, kindergarten, STEM, early childhood

Acknowledgments

I simply cannot believe how lucky I am to be surrounded by so many amazing and supportive people in my life. For that, I am truly thankful.

First, I must acknowledge my incredible committee. I still am in awe that these four amazing people came together to evaluate and support my work. An enormous Thank You to Marina Bers who has been my advisor and mentor throughout my time at Tufts. In a field where very few people are studying young children and technology, she has truly been an inspiration. Under her guidance, I was afforded so many opportunities to explore my own research interests and grow as a scholar. I cannot thank her enough and look forward to continued collaborations.

Of course, this dissertation would not exist without my unbelievable dissertation committee members, Richard M. Lerner and Megan McClelland. Rich - Thank you for your mentorship, guidance, and many edits over the years. Megan – I am so thrilled you took a chance on a student across the country that you barely knew. I am thankful for your advice, guidance, and perspective on my dissertation work. Finally, to my outside reader, Karen Brennan, whose own work and career I look up to. Thank you for your nuanced assessment of my work from our similar backgrounds and for continuing to be a friendly and supportive part of the extended Scratch/ScratchJr family.

A huge Thank You to the DevTech Lab, especially Amanda Sullivan, Amanda Strawhacker, and Dylan Portelance, and the many, video coders and classroom assistants without whom this dissertation would have never been

possible. Thank you to Louise Flannery. We spent our first four years with our desks literally attached to one another. I cannot imagine what life at DevTech would have been like without her friendship, support, and collaboration.

I must also thank Erin Seaton and Michelle Wilkerson-Jerde who both consulted on research projects long after I left their classes, and, Maryanne Wolf who served as my second advisor for the first part of my program and supervised part of my internship work. I can't fully express how you all shaped my work but I am truly grateful for your influence.

Many many many thanks to the amazing, brilliant ScratchJr team: Paula Bonta, Brian Silverman, and Mitch Resnick.

Now, how do I possibly thank all my friend and family for everything they have done over the past five years? Life at EP would not have been the same without the other half of my cohort of two: Jen Elise Prescott. Thanks for being there through all the good and bad over the past five years and making me laugh a lot along the way. I certainly could not have done this without my writing buddies: Zara Spooner and Mariah Contreras – thank you for all the peer pressure, supportive texts and emails, Saturdays in the library, and early mornings at Diesel. I am also so happy to be surrounded by so many other intelligent, friendly, and fun doctoral students who have kept me sane over many years of “Free App Nights.” Special thanks as well, to my friend and mentor, Rachel Schechter.

Thank you to my proofreading elves (both official and unofficial) – I know this my Achilles' heel and I could not have done this without your

tremendous support and assistance.

Thank you to my aunts, Dot and Shirl, for providing me with a retreat when I needed it and for their never ending enthusiasm about everything I do – you know how to make a someone feel special.

Thank you to my friends outside the program scattered all over the country who remind me of who I am and are not afraid to put me in my place once in a while when I've been going on about my data for far too long...or have abandoned them for far longer than is reasonable. Yes, I need that, and I love you all for it.

To all my friends in Boston, especially Jen who I met the first day of analyst/buyer training at TJX and has been reminding me ever since why I'm a much better academic scholar than shoe buyer – thank you!

Miro - perhaps the greatest gift we gave to each other's success was finding the strength to leave each other alone. In the end, I couldn't have done this without you, in whatever way we define that. Thank you for being such a supportive friend over the past few months. Your advice and unwavering support, in spite of everything, has meant the world to me.

To my brother Rich, and sister-in-law, Jen, thank you for being there to geek out with me about technological gadgets, education, coding, and all things nerdy. Thanks too for raising the most intelligent, adorable, hysterical little nerdlets a proud aunt could ever ask for – both as a niece and nephews and occasional beta testers.

To my sister, Dani, who always seems to be the most proud of me no

matter what. Thank you for being there with late-night texts of laughter and reminding me that occasionally I need to leave the house and have fun...preferably in clothes I didn't own in high school.

Finally, to my mom, Trisha, for always being my rock. I cannot imagine how I would have done this without your help. Your strength and perseverance has always been the model for my drive and ambition. I just wish I could be a fraction as giving as you are to others. Thank you for everything. This work is dedicated to you.

The material in this dissertation is based upon work supported by National Science Foundation DRL-1118664. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the author and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

Abstract	ii
Acknowledgments	iv
Chapter One: Introduction	2
Statement of Problem	13
Research Questions	18
Chapter Two: Review of the Literature	20
Developmental Perspectives	20
Coding Tools for Young Children	35
Chapter Three: Developing ScratchJr	45
Designing ScratchJr: Software	45
Designing ScratchJr: Curriculum	48
Chapter Four: Method	53
Design-Based Research	53
Sample	54
Procedure	58
Measures	62
Chapter Five: Results	72
BRIEF Scores	73
Research Question One	75
Research Question Two	83
Chapter Six: Case Studies	86
Chapter Seven: Discussion	103
Limitations, Future Directions, and Conclusions	109
Summary of Future Directions	115
List of Appendices	122
Endnotes	126
References	127

List of Tables

TABLE 1. BRIEF - HTKSPRE CORRELATIONS	73
TABLE 2. BRIEF - HTKSPPOST CORRELATIONS	73
TABLE 3. BRIEF - PROGRAMMING SCORE CORRELATIONS	74
TABLE 4. BRIEF - TIME ON TASK CORRELATIONS	74
TABLE 5. BRIEF - GOAL COMPLETION SCORE CORRELATIONS	74
TABLE 6. SELF-REGULATION – PROGRAMMING SCORE CORRELATIONS	77
TABLE 7. INITIAL SELF-REGULATION - GOAL COMPLETION SCORE LESSON CORRELATIONS	78
TABLE 8. SELF-REGULATION – TIME ON TASK SCORE CORRELATIONS	78

List of Figures

FIGURE 1. SCRATCHJR MAIN SCREEN	47
FIGURE 2. CHARACTER AND TEXT SELECTION	47
FIGURE 3. MOTION BLOCKS	47
FIGURE 4. GRID FEATURE	48
FIGURE 5. LESSON THREE.....	49
FIGURE 6. PROCEDURE TIMELINE (75 DAYS TOTAL).....	60
FIGURE 7. CAMTASIA SCREEN	64
FIGURE 8. DISTRIBUTION OF HTKS PRE-TEST SCORES	72
FIGURE 9. TIME ON TASK PERCENTAGE BY LESSON	78
FIGURE 10. PROGRAMMING SCORE AND SELF-REGULATION SCATTERPLOT	80
FIGURE 11. GOAL COMPLETION SCORE AND SELF-REGULATION SCATTERPLOT WITH PROJECTS	81
FIGURE 12. AVERAGE TIME ON TASK PERCENTAGE AND SELF-REGULATION	82
FIGURE 13. AVERAGE TIME ON TASK PERCENTAGE AND SELF-REGULATION FOR PROJECTS	83
FIGURE 14. JACK’S AIRPLANE LESSON SCRATCHJR PROJECT.....	90
FIGURE 15. THE END STATE OF SARAH’S RACE PROGRAM	90
FIGURE 16. SCREENSHOT OF SARAH’S RACE PROJECT WITH COLORED CHARACTERS	92
FIGURE 17. EXAMPLES OF CATS IN SPACE SCRATCHJR PROGRAMS.....	93
FIGURE 18. PARALLEL PROGRAMMING.....	96

Cats in Space, Pigs that Race: Does self-regulation play a role when kindergartners learn to code?

Chapter One: Introduction

When work on this dissertation began, organizations that promote learning to code (i.e., programming a computer's actions), such as Code.org (<http://www.code.org>), Code Academy (<http://www.codecademy.com/>), Code to Learn Foundation (<http://codetolearn.org>), and Girls Who Code (<http://www.girlswhocode.com>), to name a few, did not exist. However, there has recently been a renaissance of sorts in the "learning to code" movement that has not been felt since the early 1980s, when computer programming in schools was fueled by the prior integration of computers into the classroom, the invention of programming language for elementary-aged children called LOGO, and an influential book by Seymour Papert (1980), *Mindstorms*. While learning to code is gaining a renewed popularity in K-12 education, there are limited resources and research targeting early childhood technology and engineering education. The work in this dissertation aims to fill in part of that research void.

A topic more heavily explored in the field of early childhood development and education is that of self-regulation. Broadly, self-regulation refers to an integrative process that governs emotional, cognitive, and behavioral functioning (Baumeister & Vohs, 2004; Gestdotteir & Lerner, 2008; McClelland, Ponitz, Messersmith, & Tominey, 2010). More specifically, the work in this dissertation focuses on measurable behavioral self-regulation including working memory, attention, and inhibitory control (McClelland, et al., 2007; Ponitz et al., 2008). There is a particular focus on self-regulation in early childhood, as it is believed to be key in the transition-to-school years (Diamond, 2002; Sameroff & Haith,

1996). There is an emphasis in this work on the connection between self-regulation and learning to code as coding is problem-solving (see Brennan & Resnick, 2012; Yelland, 2005 for descriptions of problem-solving with Scratch/LOGO) and problem-solving is a self-regulative process (Zelazo, Carter, Reznick, & Frye, 1997).

As this work is grounded in the field of early childhood development, this dissertation will attempt to connect self-regulation and learning to code during this period of development. The software tool used in this study, ScratchJr, was developed with early childhood developmental theory in mind, paying particular attention to young children's self-regulation, early math ability, and early literacy skills. For example, the number of coding blocks children are presented with in the ScratchJr programming language is limited due to children's developing attention and working memory. There are very few words in ScratchJr due to developing literacy skills, although children can add text to practice these skills. Almost no "instant gratification" buttons (blocks you can click for an immediate action or reward) exist in ScratchJr. The removal of these "instant gratification" buttons is due to young children's developing inhibitory control, in order to encourage programming over simply clicking for cause and effect. The number range children can work with in ScratchJr is limited to 20 horizontally and 15 vertically based on young children's number sense and inhibition control (i.e., a limit on how large a number can be input into the software).

ScratchJr can trace its lineage back to the LOGO programming language. What is particularly unique and exciting about ScratchJr is that it is specifically

designed for children ages five to seven, building off of a rich coding/computer programming history, but also integrating foundational knowledge of early childhood development into its creation. The ScratchJr programming language was developed in conjunction with collaborators of Seymour Papert (LOGO) and the developers of the original Scratch graphical programming language (see: Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010; Resnick, Maloney, Monroy-Hernandez, Rusk, Eastmond, et. al., 2009 for descriptions of the creation of Scratch).

Learning to code could mean many different things, depending on context and programming language. As noted, a high-level definition of coding is programming a computer's actions. For the sake of this dissertation, when using an early childhood graphical programming language, learning to code means selecting a character, selecting the appropriate programming blocks (pieces of code), and stringing those pieces of code together, in a correct sequence, in order to make the selected character or characters perform the goal actions, whether that goal is set by the child or an outside source.

At this writing, it is exciting to witness the resurgence of the "Learn to Code" movement. The resurgence is demonstrated by foundations devoted to the cause, monthly news articles about the topic, weekly organized "Twitter chats," and an "Hour of Code" during Computer Science Education Week (December 9 – 15, 2013) that got over 23 million children and youth to learn an hour of code (<http://www.csedweek.org>).

Typically, discussions about learning to code involve how to teach older children and adolescents how to code to better prepare them for the job market. Even some of the recent discussions around young children learning to code come with the same advertisement of a promising financial future (Huebner, 2013). With ScratchJr, the ability to learn to code is available to young children as young as five. Learning to code with ScratchJr is more about the fundamental educational and personal growth benefits that come with learning to think computationally, rather than with thinking about specific career pursuits. The goal of teaching young children to code with ScratchJr is to build creative, computational thinkers; engaging children in skills that may translate across classroom disciplines and life skills.

Jeannette Wing (2006) argued that every child should learn computational thinking, or thinking like a computer scientist (e.g., thinking logically and systematically through the use of digital technology), in addition to reading, writing, and arithmetic. The crux of this argument was that thinking like a computer scientist allows children to logically break down any problem he or she may encounter. There seems to be value in teaching the critical thinking skills and problem-solving mindset that come with thinking like a computer scientist, along with traditional reading, writing, and arithmetic. In fact, due to the ubiquitous nature of digital technology in our society, some researchers argue that 65% of the careers young children will have in the future have yet to be invented because they will be based on the needs of this ever-changing digital landscape (Davidson, 2011). Preparing for the job market, however, is not the only reason

young children should learn to code – computational thinking, problem solving, and critical thinking skills are relevant skills throughout school and life.

In addition, it may be beneficial to begin teaching digital literacy skills, including coding, in early childhood, as long as there are developmentally appropriate technologies available. Learning to code, similar to learning to write, provides children with a means of self-expression. However, learning to code also affords children an additional understanding of their digital worlds through learning the basic digital building blocks of the digital technologies around them. Furthermore, learning to code, unlike learning to write, provides immediate feedback about right and wrong syntax and immediate cause-and-effect interactions are on display for children to watch, manipulate, and redesign. Using Wing's terminology (2006), "thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction (p. 36)." When young children learn a new programming language, they are not "just learning to code, they are coding to learn (Resnick, 2013, para 5)."

Teaching children to code in early childhood, rather than waiting until later elementary school, builds on a history of early childhood interventions with long-term economic payoffs. Investing in interventions in early childhood is associated with lower costs, and longer-lasting effects than interventions that begin later in childhood (Heckman, 2006; Heckman & Masterov, 2004; Reynolds, et al., 2011). Even though self-regulation is highly predictive of later life success, many children arrive at school without adequate self-regulation skills (Blair,

2002). Leveraging the deficit in self-regulation skills through interventions is particularly important for and beneficial to economically disadvantage families (Cunha & Heckman, 2007). Research also suggests that children who are exposed to STEM (science, technology, engineering, and math) curriculum and coding skills at an early age demonstrate fewer gender-based stereotypes regarding STEM careers (Metz, 2007; Steele, 1997) and fewer obstacles entering these fields (Madill, et al., 2007; Markert, 1996) if a STEM career becomes the child's eventual goal.

It seems appropriate to teach all aspects of STEM, including a new emphasis on the T and E, in early childhood. Young children are budding scientists and engineers, acting as investigators on the playground – frequently exploring the newest objects within their reach, constantly asking questions, and experimenting with new materials (STEM Smart Brief, 2013). There is focus on math in early childhood but with little focus on the S (science), and almost no focus on the T (technology), and E (engineering) in PreK – 3rd grade (STEM Smart Brief, 2013).

Part of science learning is hypothesis testing, which is, part of the scientific method or engineering design process (see <http://www.eie.org/overview/engineering-design-process> for a child-friendly version). Executive function (EF) plays a role in a child's capacity to iteratively revise a hypothesis (see page 28 for a discussion of the connection between self-regulation and EF). A child can still engage in testing a hypothesis at lower levels of EF development, even if the ability to predict and revise a hypothesis has not

yet developed (Gropen, Clark-Chiarelli, Hoisington, & Ehrlich, 2011). Testing and revising hypotheses is a key component of not only the scientific method but this process is also integral to testing and debugging in the engineering design process while learning to code. If children can test hypotheses before they can revise and confirm them, they can still engage in meaningful science and engineering exploration in early childhood and work on building EF skills through scaffolded scientific inquiry (Gropen, Clark-Chiarelli, Hoisington, & Ehrlich, 2011).

Several studies have explored the relationship between math and self-regulation in early childhood. Inhibition control was central to preschool math skills even when shifting and working memory were controlled (Espy, et. al., 2004). Bull and Scerif (2001) identified inhibition and working memory deficits as factors in math difficulties among six to eight year olds. One intervention, which combined mathematics (Building Blocks) and self-regulation (Tools of the Mind) training in preschool classrooms, however, did not produce the predicted increase in self-regulation skills from the combined curriculum units. Instead, the authors claim, there may have been a small increase in self-regulation from the Building Blocks curriculum alone (Clements, Sarama, Unlu, & Layzer, 2012). However, this increase from the curriculum could be due to the above-mentioned connection between math skills and self-regulation (McClelland, et. al., 2007).

In regard to technology education, Gee (2008) argued that, by fourth grade, the digital gap compounds with the “4th grade slump” in traditional literacy, each making the other worse. This relation means that children who

struggle in the transition from “learning to read” to “reading to learn” also tend to be the children who have not developed 21st century digital skills and have less exposure to digital media and tools (Gee, 2008). “The fourth-grade slump consistently leads to educational failure. The digital gap leads to a failure to become confidently ‘tech-savvy,’ a 21st-century skill crucial for success, and even for survival (Gee, 2008, p. 4).” The combination of these two deficits is a potentially dangerous setback for children in the 21st Century.

Building on the importance of STEM education and teaching digital literacy and 21st century skills (including critical thinking, problem solving, communication and collaboration skills along with information, media, and technology literacy (P21, 2009)), the ScratchJr programming language was created in order to develop an educational technology tool specifically tailored for early childhood that is rooted within developmentally appropriate practice. Developmentally appropriate practice (DAP) means fostering learning environments sensitive to children's social, emotional, physical, and cognitive development needs (Copple & Bredekamp, 2009). One particular area of interest in early childhood development, as it is a DAP curricular goal (Copple & Bredekamp, 2009), is self-regulation, which is the focus of this dissertation.

Self-regulation is described and defined from various perspectives and measured in a myriad of ways (McClelland, Pontiz, Messersmith, & Tominey, 2010). The *Handbook of Self-Regulation* (Baumeister & Vohs, 2004) states that all contributing authors had different definitions of self-regulation but could agree on a common theme: “self-regulation refers to the exercise of control over

oneself, especially with regard to bringing the self into line with preferred (thus, regular) standards (Vohs & Baumeister, 2004, p. 2).” Self-regulation is “people regulating their thoughts, emotions, impulses or appetites, and task performances...[and] attentional processes (Vohs & Baumeister, 2004, p. 2).” For this study, self-regulation will be evaluated from primarily an education perspective, as the research conducted in this study took place within the context of kindergarten classrooms. An education perspective of self-regulation examines behaviors, thoughts, and feelings associated with school success (McClelland, Pontiz, Messersmith, & Tominey, 2010). “Self-regulation generally refers to the capability of controlling or directing one’s attention, thoughts, emotions, and actions (McClelland & Cameron, 2012, p.136)” and is defined in this dissertation by the behavioral aspects of self-regulation: flexible attention, working memory, and inhibitory control (McClelland & Cameron, 2011). This definition is used because it fits with the general consensus in the field that self-regulation refers to “controlling, directing, and planning cognitions, emotions, and behaviors (Baumeister & Vohs, 2004; McClelland et. al., 2010, Schunk & Zimmerman, 1997) (McClelland & Cameron, 2011, p. 32).” In addition, the separate yet integrated tasks of flexible attention, working memory, and inhibitory control appear to be most relevant for learning in school, particularly when following directions or persisting on difficult tasks (McClelland & Cameron, 2011).

Of course, children may have variations in their approaches to ScratchJr and their self-regulation skills. Individual children have independent, developing personalities that influence how they approach novel environments in school and

at home. Children differ in their displays of emotion, activity, and attention (Rothbart, Sheese, & Posner, 2007). This development of personality is sometimes referred to as temperament, which is the individual variation in children's reactions to the environment (Rothbart, 2007). A late-emerging factor of temperament is effortful control, which includes the inhibitory control and attentional focusing and shifting components of self-regulation (Rothbart, Sheese, & Posner, 2007).

Self-regulation, as noted, is important to school success. It predicts school readiness over and above other factors, such as general cognitive skills and family background (Blair & Razza, 2007). Behavioral regulation scores (using the same assessment as the present study) accurately predicted literacy and math skills in prekindergarten students (McClelland, et. al., 2007). In addition, growth in behavioral regulation predicted growth in math and literacy skills (McClelland, et. al., 2007).

This dissertation builds on the importance of STEM education, teaching digital literacy and 21st century skills, the predictive nature of self-regulation for academic success, and the recent resurgence of focus on “learning to code” tools for children. The study reported herein was designed to understand one kindergarten cohort's varied achievements in learning the ScratchJr programming language as part of the ScratchJr project, an NSF-funded collaboration between MIT's Lifelong Kindergarten Group, Tufts University's DevTech Lab, and The Playful Invention Company (PICO).

As a whole, the ScratchJr project introduced a developmentally appropriate computer programming language (Flannery, Kazakoff, Bontá, Silverman, Bers, & Resnick, 2013), ScratchJr, and corresponding curriculum units to children aged five to seven. When ScratchJr was pilot tested, the design of the software was based on prior observations of children using other developmentally appropriate (and non-developmentally appropriate) programming tools (e.g., Scratch, CHERP, Lego WeDO). Observations were made, and pilot studies were conducted, to determine what design features would be included in the first design of ScratchJr in order to accommodate the varying levels of self-regulation in children aged five to seven (Flannery, et. al., 2013).

A hypothesis of the ScratchJr project overall was that introducing programming tools into the context of an early childhood classroom might not only spark children's interest and curiosity in digital technologies in the early years, but also build skills and knowledge useful across multiple domains, including literacy, mathematics, problem-solving, self-regulation, and 21st century skills. The intention of the ScratchJr programming tool is that it will be integrated as a digital tool for learning computational thinking and digital literacy skills in the early childhood in conjunction with the traditional early childhood curriculum. Early childhood teachers, and kindergarten to second grade students, were influential in the design of the software and accompanying curriculum from the bottom up.

Working within the context of the ScratchJr project, this dissertation explored, in two kindergarten classrooms, the potential relationship between self-

regulation and children's experiences in learning to code using the ScratchJr programming language within the context of their own classrooms. Specifically, this dissertation examined if initial levels of self-regulation made a difference in ScratchJr performance during the first eight ScratchJr lessons, and if varied lengths of exposure to ScratchJr correlated with changes in self-regulation scores. Analyses in these areas were conducted to examine if there was a potential relationship between self-regulation and learning to use ScratchJr.

Statement of Problem

Technology education and early childhood education are not always strongly linked. There are very few technology tools designed for use specifically in early childhood and little research about understanding the use of digital devices in early childhood or how to design developmentally appropriate technological tools for young children. Drawing on five interrelated questions developmental scientists may ask themselves when studying development from an relational developmental systems perspective (Jelicic, Theokas, Phelps, & Lerner, 2007; see page 20 for a detailed explanation), I propose the following, general question to frame my work, "what developmental and personal attributes of young children, in the context of home and school, during the 21st Century, promote positive child development when using digital tools?" This dissertation aimed to be a stepping-stone towards creating and integrating developmentally-appropriate technological tools into early childhood classrooms, taking into account a fundamental aspect of early childhood development – self-regulation – and a specific area of technological knowledge – coding. In the general study of

ScratchJr, additional areas were and will continue to be assessed, including connections to math, literacy, problem solving, curriculum design, and collaboration among students. However, the specific focus of this dissertation work was self-regulation. Therefore, building on the general framing question above, the more specific, overarching question for this dissertation study was “to what extent does self-regulation have a role in learning to code with a novel computer programming software in kindergarten classrooms, and, does length of exposure to the programming software correlate with post-test self-regulation scores of these kindergarten students?”

The term coding is used instead of programming to fit with the contemporary literature and colloquial references around teaching concepts of computer science for K-12 (see <http://www.code.org>; <http://www.codetolearn.org>; <http://www.csta.org>). However, coding and programming will sometimes be used interchangeably in this manuscript in order to avoid confusion around “coding tools” for qualitative and quantitative analysis and the act of “learning to code” with ScratchJr. The process of learning to code can sometimes be thought of as putting pieces of code together in order to make a program for a computational device to read and execute.

To achieve the goal of this dissertation, which was to explore the possible bidirectional relationship between self-regulation and coding skills in kindergarteners, a graphical programming tool in development specifically for young children (ages five to seven), ScratchJr, was used. Children’s self-regulation skills were assessed before and after using the ScratchJr software and

associated curriculum. The focus of this work was not just to understand how to develop software tools for the best performing, most attentive, or most motivated children. Instead, this dissertation aimed to understand how differences in the development of self-regulation skills and learning to code with a new software tool for young children might be related (see page 51 for specifics of how self-regulation was accounted for in the design of ScratchJr.).

Purpose

Over multiple years of working with young children and digital tools, it became clear to me there was something at play beyond a child's intelligence, math skills, literacy skills, family background, etc. that determined how well he or she learned the software or hardware presented. When working with children in prior studies, some children meticulously attended to the task at hand while others ran up and down the stairs, added hundreds of characters to the screen, or built towers out of Lego.

Ideally, in building developmentally appropriate digital tools for young children, epistemological pluralism should be taken into consideration. For example, in the DevTech Research Group's work with robotics, children create personally meaningful robotic artifacts, such as ballerinas, cats, or monster trucks. Epistemological pluralism, specifically when using computational tools, speaks to multiple modes of entry to computation (Turkle & Papert, 1990). Even though the computer is thought of as a mathematical or logical device, some children will approach the tool with a more artistic orientation (Turkle & Papert, 1990). There

are many diverse approaches to programming and building a developmentally appropriate digital tool means accounting for this epistemological pluralism.

I am not ignoring epistemological pluralism when I speak of the limitations of young children working with Scratch or children running up and down the stairs instead of programming robots. Instead, my intention is to highlight the need for understanding what may be barriers of entry to even approaching new digital tools. If a digital tool is developmentally inappropriate, if the floor is not low enough, we cannot even begin to speak of diverse approaches to learning within the digital environment since the child is experiencing a barrier to entry.

This barrier to entry phenomena was particularly apparent when working with young children (five to seven year olds) using the original version of the Scratch software (for children ages 8 and older). The kindergartens seemed to have a difficult time getting passed “instant gratification” type buttons (i.e., ones that would randomly insert characters or make a character grow big and shrink small) in favor of systematically programming their great ideas. This dissertation sought to explore, describe, quantify, and analyze those differences in self-regulation, on a small pilot scale, by examining the possible bi-directional relationship between self-regulation and learning to code with ScratchJr. Once a developmentally appropriate programming software is realized, additional work can be conducted on individualize approaches to learning to code.

Why coding education? Take, for example, Douglas Rushkoff’s concept of “Program or Be Programmed” or James Paul Gee’s “tech insiders” vs. “tech

insiders” – both of these ideas speak to the idea of those who know how digital machines work versus those who just use the digital machines (Gee, 2013a; Rushkoff, 2010). While these may be somewhat extreme views, the underlying sentiment holds true as a motivation for this dissertation work. Young children grow up surrounded by digital devices and yet know very little about how these tools work (Bers, 2008). This situation is especially true for children from families with lower socioeconomic status whose families have less exposure to new digital devices on a daily basis (Gee, 2013a; 2013b, Gutnick, Robb, Takeuchi, & Kotler, 2010). In addition, very few developmentally appropriate educational technology tools exist specifically designed for young children. Understanding how different levels of self-regulation may influence a child’s ability to navigate a digital tool and learn to code can inform the design and evaluation of digital tools for young children in order to ensure the most universal access and appeal and most successful use of digital devices possible for all children.

Furthermore, understanding if learning to code can positively influence self-regulation skills (and, in future studies, math and literacy skills), is potentially an argument for integrating computer science education into early childhood classrooms as a method to teach digital literacy skills in a way that enhances, rather than distracts from, the standard curriculum. Teaching with an integrated curriculum, rather than one that separates STEM from social and emotional skills in the realm of self-awareness, social-awareness, self-control, and relational skills, could potentially create more well-balanced and well-prepared citizens for the 21st

century, which will likely need collaborative, creative, problem-solvers both in school and in future careers. “The workplace and the classroom increasingly require ready access to information and analytical and creative thinking skills that allow for self-regulated learning through goal setting, strategy use, and self-monitoring (Blair, 2002, p. 111).”

Research Questions

In order to further understand if a technological tool could be integrated into kindergarten classrooms, and provide a developmentally appropriate means to study self-regulation skills along with digital literacy skills, this dissertation focused on the following specific questions within the larger scope of the ScratchJr project: (1) To what extent do differences in initial levels of self-regulation of kindergarten-aged children account for differences in ScratchJr performance? (2) To what extent do differences in exposure to the ScratchJr software correlate with differences in post-test self-regulation scores in kindergarten classrooms?

Keep in mind that when discussing ScratchJr integration in the classroom, ScratchJr does not refer to simply the software tool. The tool was not left alone without teachers or curriculum. There are many interacting environmental variables that come into play in a child’s world that influence his or her development. To further frame these research questions, the next chapter will outline the literature related to young children’s development, self-regulation, and learning to code. Subsequent chapters will describe the research methodology, results, conclusions, and future directions of the work carried out in examining the

relationship between self-regulation and learning to code in kindergarten classrooms.

Chapter Two: Review of the Literature

This chapter provides an overview of background literature pertaining to relevant child development perspectives, self-regulation, and coding for young children.

Developmental Perspectives

ScratchJr is an interdisciplinary project. As such, this work integrates several areas of developmental theory, which are outlined in the following sections.

Relational Developmental Systems Theories

Relational developmental systems theories (RDST) are viewed as the over-arching theory meta-theory for the work presented here. RDST takes into consideration a child and his or her whole context, with all factors of a child's life considered: biological, cultural, historical, environmental, social, etc. and interactions between these factors (Lerner, 2006; Overton, 2010; Overton & Mueller, 2012). The child is viewed as an active agent in his or her own regulation and this theoretical orientation is particularly useful when taking into account children in the context of their classroom settings (McClelland & Cameron, 2011).

In RDST the individual-context relation is the primary unit of analysis when studying human development from this perspective, as the bidirectional interactions between the individual and contexts is the basic process of development (Lerner, 2002). The concepts underlying RDST are key for understanding children interacting with a novel tool, such as learning to code with

the software tool, ScratchJr, within their classroom contexts. Looking at the use of ScratchJr from an RDST perspective takes into consideration interactions between teachers, peers, and researchers, while these children are learning a new skill within the context of a novel environment (the programming language) and a familiar environment (their classroom) among novel researchers and familiar teachers and peers. The nature of classroom research is one in which a child cannot be split from the sociocultural influences the context may have on the child, and, in turn, the child may have on the environment. As such, the research questions in this dissertation, are bidirectional in nature: does the child's initial self-regulation correlate with coding skills in the ScratchJr programming language (child → context) and does length of exposure to the ScratchJr programming language correlate with post-test self-regulation scores (context → child)?

Another key component of RDST is the idea that there are multiple possible ideal trajectories of human development – developmental trajectories are relatively plastic in nature (Ford & Lerner, 1992). The coaction of genetics and environment shape the constraints on one's development, but this shaping varies from person to person (Ford & Lerner, 1992). This idea is important when studying learning trajectories in classrooms, as all children will not follow the same paths even within the same context. To further highlight this point, there are five interrelated questions developmental scientists may ask themselves when studying development using RDST; these questions are particularly applicable to studying early childhood development with technology. They are (Jelicic, Theokas, Phelps, & Lerner, 2007): “(1) What attributes of (2) what individuals?

in relation to (3) what contextual/ecological conditions? at (4) what point in time? may be integrated to promote (5) what instances of positive human development (p. 10)?”

These questions highlight some of the inquiries one would make when studying child development and technology, for example: “what developmental and personal attributes of young children, in the context of both home and school, during the 21st Century, promote positive child development using technological tools?” Or, more specifically for this dissertation, “to what extent does self-regulation have a role in learning to code with a novel computer programming software in kindergarten classrooms, and, does length of exposure to the programming software correlate with the post-test self-regulation scores of these kindergarten students?”

Positive Technological Development (PTD)

The study of positive youth development (PYD) emerged from RDST, and focuses on positive developmental attributes of adolescents (Lerner, Lerner, Bowers, & Geldhof, in press). Bers (2007; 2012) developed the positive technological development framework (PTD) from the positive youth development (PYD) perspective and Papert’s (1980) *constructionism* (described in more detail below) (Bers, 2007; 2012). The core questions of PTD ask “How can children use technology in positive ways to help themselves and the world?” and “How can educators and researchers develop programs that help children use technology to learn new things, to express themselves in creative ways, to

communicate effectively, to take care of themselves and each other, and to contribute in positive ways to the self and the world” (Bers, 2007; 2012).

While PYD considers the assets that typically lead to positive contributions youth make to society, PYD does not necessarily account for the influence of technological tools. PTD attempts to bridge this gap (Bers, 2012). Both PTD and PYD emphasize six positive attributes: caring (human values, empathy, social justice), connection (positive bonds with others), contribution (civic engagement), competence (cognitive abilities and orientation towards healthy behaviors), confidence (self-efficacy and positive self-regard), and character (integrity and morality) (Bers, 2012; Lerner, et. al., 2005). PTD adds to PYD and connects these “C’s” to corresponding action-oriented “Cs” that are developed by engaging with new technologies: communication (exchanges of ideas), collaboration (engaging in shared projects), community-building (especially in virtual spaces and social networks), content-creation (creation of digital content that fosters technological fluency), creativity (using a variety of multimedia approaches to express oneself and solve problems), and conduct (ethical and moral choices in virtual spaces, social networks, and with limited supplies of digital tools) (Bers, 2007; 2010; 2012).

These 12 “Cs” are particularly useful to consider when developing software, hardware, and accompanying curriculum for young children, to ensure the technology tools are being used in developmentally appropriate ways.

Technology designers may consider how to build a tool that particularly fosters

PTD in the user. For example, does the tool foster creativity? Can the tool be used for self-expression or exploring novel approaches to problem-solving?

In sum, both PYD and PTD, which derived from PYD, draw upon RDST. PTD and PYD are useful for studying the child in context and keeping in mind attributes that typically lead to positive developmental outcomes. PTD is especially beneficial to draw upon when understanding how children make use of technological tools.

Constructivism/Constructionism

Another developmental perspective drawn upon for the work presented here is Jean Piaget's constructivism. As in RDST, constructivism is the idea that children learn by interacting with their environments, they are not just passive consumers of knowledge, and that all knowledge is constructed (Forman, 1988). Constructivism is mentioned both as a foundational theory in the study of technological tools for children (see Forman & Pufall, 1988) and as a reference point for the discussion of Seymour Papert's constructionism.

Seymour Papert, a pioneer in the field of children and technology, extended Piaget's idea of "constructivism" into the term "constructionism" to emphasize learning by making physical constructions, representations, and designs (Papert, 1980; 1991). Papert emphasized constructions in the digital space, inventing the LOGO programming language for young children (Papert, 1980; 1991). Programming languages for children, starting with LOGO, come from a history of constructivist pedagogy as Papert was a colleague of Piaget's

and derived his theory of constructionism from constructivism (Forman, 1988; Papert, 1980; 1991).

Building on Piaget's constructivist ideas are also project-based/inquiry-based learning models that provide more scaffolding than the traditional view of constructivist pedagogy, which is viewed as having minimal guidance or structure (Hmelo-Silver, Duncan, & Chinn, 2007). All learning models essentially build on constructivist approaches as all learning involves the construction of knowledge; however, additional scaffolding is provided in the problem-based approach (Hmelo-Silver, Duncan, & Chinn, 2007). Students are provided with guidance, structured instructions, and hints, but are not given the answer to the problem, making difficult tasks more achievable (Hmelo-Silver, Duncan, & Chinn, 2007).

The ScratchJr curriculum uses a similar approach to the project-based/inquiry-based model that assigns students challenges and demonstrates a task to be carried out. Various curricula were assessed with different levels of scaffolding throughout the larger-scale ScratchJr project. The children included in the version of the ScratchJr study reported here used the first version of the most open-end, least-scaffolded, project-based curriculum (Flannery, et. al., 2013). Additional details on the curriculum can be found throughout this dissertation, with particular focus on the potential limitations in the Case Study and Discussion Chapters.

In sum, Papert built on Piaget's work to bring the idea of constructivism to working with digital tools. Both Papert's and Piaget's theories are influential on software and curriculum development, including the software and curriculum

designed for the ScratchJr project. The work of Lev Vygotsky, highlighted in the next section, adds a social-cultural perspective to the foundational theories of developmental psychology, a perspective that many believed Piaget's theories often lacked (Piaget, 2000).

Social Constructivism

Lev Vygotsky's (1978) ideas were in many ways similar to those of Piaget in regard to learning, but stressed the social context of learning as important. Vygotsky argued things as well as people were socially constructed (e.g., symbolic objects) (Scribner, 1990). This idea is particularly relevant when learning a programming language that is symbolically based.

Vygotsky also stressed the importance of the role peers, teachers, and caregivers had on a child's cognitive development and learning. Relevant for the study of self-regulation in the classroom environment and for the study of digital technology's impact on children's lives, Vygotsky (1978) presented the idea that individuals are embedded within an environment and the environment and individual interact with each other, ideas found also in RDST and Piaget (1970). Illustrative of this point is the idea that technological tools are created by the culture within which the child lives. All technological tools contain symbolic artifacts and are mediated by social groups and cultural mores (Moll, 2014). Younger children do not provide themselves with the digital technologies in their lives; parents, families, and schools are the ones to make the purchases or hand the child the devices (Gutnick, A. L., Robb, M., Takeuchi, L., & Kotler, J., 2010).

Therefore, the technological tools the child is exposed to are influenced by societal and cultural factors.

For young children who are in a developmental process of learning how to work with others, the design features of the technological tools may promote pro-social development through collaboration with other children while using new technologies. New technologies might help to foster interactions between peers who may otherwise be focused on their own thoughts and perspectives (egocentrism) and might also engage children in partnerships that expand the child's zone of proximal development (Vygotsky, 1978). For example, a child who is better skilled at using the mouse or browsing the web might work together with a child who had less exposure to these computer tasks; a child who has used a digital camera at home, might show another child which button to press; and children may show each other their favorite smartphone apps and instruct each other on initial play instructions or create their own off-screen games to mimic those found on-screen. In addition, with ScratchJr, coding a character on screen may provide children with a different perspective view from their own.

Research shows that there is more spontaneous peer teaching and helping at a computer screen than during other classroom activities (Clements & Nastasi, 1992). Digital tools, and understanding how to use a variety of digital tools, may provide additional means for fostering these social-emotional skills in early childhood classrooms and at home as well as provide additional pathways and motivation for collaboration, communication, and self-expression.

In sum, this interdisciplinary dissertation draws on ideas from RDST, constructivism, constructionism, social-constructivist theory, and frameworks such as PTD and PYD. The strengths of these perspectives come from: (1) acknowledging child-environment interactions; that (2) the child has some control over his or her own learning within a context; and that (3) the context is influenced by the culture and society in which the child lives and is raised. These points are all essential for understanding how a child learns to use, and self-regulates around using, new digital tools. Given the theoretical background outlined above, the next section will discuss more in depth the concept of self-regulation in the context of early childhood development.

Self-Regulation

My analysis of developmental theories emphasized the role of the active child in the developmental process, drawing on my work in environments that stress constructivism, constructionism, and RSDT. For a child to be engaged in the constructivist/constructionist learning, the process of self-regulation is involved, as the process of learning relies more on targeted scaffolding than direct instruction.

Self-regulation and EF skills are important to development across the life span, and they are particularly important for mastering 21st century skills (defined on page 9), such as creativity and out of the box thinking, self-control - thinking before one speaks and resisting temptation; and discipline – the ability to stay on task, stay focused, and seek a long term reward (Diamond, 2010; Diamond & Lee, 2011). All of these skills are seemingly exercised when engaging in coding tasks

with ScratchJr. For example, a child may need to develop a concept (creativity), think about how to compensate when the exact character they want or need does not exist (out of the box thinking; working memory), debug a program that is not working correctly (out of the box thinking; staying focused on a task (attention)), and continue working on his or her program despite temptations to play on other areas of the computer or device, bang on the keyboard, or simply color in the paint editor (on task; on focus; seek long term reward (inhibition control)).

EF and self-regulation are related, in that EF serves as the construct that unites working memory, attention, and inhibitory control for the purposes of planning, problem-solving and goal-directed activity (Blair 2002; Blair & Razza, 2007). However, studying attention, working memory, and inhibitory control alone may ignore the more complex areas of goal setting and abstract thought. Nevertheless, it is useful to study EF in early childhood, particularly these subareas sometimes called self-regulation, since the more complex factors of EF (i.e., metacognition, goal setting) are not easily measured in young children (Garon, Bryson, & Smith, 2008; Gioia, Isquith, Guy, Kenworthy, 2000).

Self-regulation, defined as working memory, attention, and inhibition control, was the primary area of measurement in this dissertation. These three aspects of self-regulation are key for success in the early childhood classroom, that is, they are predictive of academic success. In addition, strengthening these skills could have long-term academic benefits (Blair & Razza, 2007; Diamond, 2002; McClelland, et al., 2007; McClelland, Morrison, & Holmes, 2000; McClelland & Cameron, 2012). The HTKS assessment, a focus of measurement

in this study that assesses self-regulation directly from students, indexes these three aspects of self-regulation, as demonstrated in several prior studies utilizing the measure (McClelland & Cameron, 2012; Ponitz, McClelland, Matthews, & Morrison 2009; McClelland, et al., 2007).

Components of self-regulation, such as working memory, attentional flexibility, and inhibition control (McClelland & Cameron, 2011) appear to be essential skills involved in learning to code and are therefore assessed in this study. In regard to working memory, in order to be successful with the ScratchJr programming language, children must remember their programming challenge, the programming blocks that correspond to the actions they would like for their characters, and the series of instructions that they placed on their character.

Attentional flexibility is also a key component of learning to code with ScratchJr, as children must switch between characters, backgrounds, and programs for their different characters; switch between categories of programming blocks; and switch between multiple pages within each program. Finally, inhibition control comes into play when a child needs to inhibit a dominant response in favor of a more productive one, especially in the context of the classroom experience and when learning a new task. Children, when learning to code with ScratchJr (or another program), may also be resisting the urge, for example, to bang on their keyboards out of frustration or spend a majority of their time in the paint program rather than programming their characters.

Measures of EF and Self-Regulation

In order to measure self-regulation, two measurements were used. One administered by researchers to students, the Head Toes Knees Shoulders (HTKS) and one by teachers evaluating their students (BRIEF). The HTKS purports to measure working memory, inhibition control, and attention while the BRIEF is a more comprehensive, clinical measure of multiple facets of EF including working memory, inhibitory control, and (shifting) attention plus emotional control, initiation, planning, organization of materials, and monitoring.

The prior sections provided a general developmental overview of the “average child.” The next section will highlight some differences between children that may impact learning to code before moving into the following section, which discusses coding tools for children.

Digital Natives, Digital Divide, and the Participation Gap

Multiple studies point to the relationship between self-regulation and socioeconomic status (SES). The digital divide and participation gap are two issues in STEM education also related to SES. Although race/ethnicity and SES status were not variables specifically assessed in this dissertation at the individual child level; the school that is involved in this study is considered a high needs school and race/ethnicity and SES will be discussed as future directions for this work.

Children born in the 1990s are the first generation of children with access to new technologies, such as smart phones and YouTube, from birth. They will grow up with email and social networking in their pockets. The term “digital

natives” was coined to describe these children – those growing up in a world surrounded by new digital technologies that will never know what life was like before the Internet, not having the ability to pause television, or GPS navigation to prevent their parents from getting lost (Oblinger, 2003; Prensky, 2001; Tapscott, 1998). No generation has yet lived their whole lives in this digital culture (Palfrey & Gasser, 2008). Although this term is commonly used, it is debated. Just being born into a generation surrounded by digital technology does not mean you know how to use technological tools automatically.

However, on average, children today spend more time with digital media than engaged in any other activity other than sleeping (Roberts & Foehr, 2008). In just a few decades (the World Wide Web debuted in 1991) billions of people worldwide have invented and adopted digital technologies—hundreds of years faster than the printing press. A large body of interdisciplinary research on technology and early childhood has been conducted in the past three decades. Efforts focused on the impact of technology on children’s cognitive and academic development (Fletcher-Flinn and Gravatt, 1995; Clements, 2002) and socio-emotional development (Crook, 1998; Medvin, Reed, Behr, and Spargo, 2003; Muller and Perlmutter, 1985; Shade, Nida, Lipinski, and Watson, 1986; Wang & Ching, 2003). “Digital natives” social interactions are fundamentally changing due to the use of digital technologies. They do not know a time when it was common practice to send a handwritten note rather than an e-mail or call someone up for a date, rather than texting or Facebook-messaging them. To the digital

natives, the “real” world and the “online” world are one and the same (Palfrey & Gasser, 2008).

Although the term “digital native” (Prenkley, 2001) is widely used, it is not universally applicable. Children from lower income households are historically at a greater disadvantage compared to their peers in terms of accessing new technological tools. In addition to more limited access to expensive digital devices, children of lower income households typically have less parental influence when choosing which digital media to access and, therefore, may be less likely to choose educational websites or application (“apps”) (Gutnick, Robb, Takeuchi, & Kotler, 2010). Although some studies estimated between 70% and 80% of households with children ages birth to eight year olds have computers (Common Sense Media, 2011), the range actually varies by income, with around 48% of families who make less than \$30,000 a year owning a computer, to 91% of families who make over \$75,000 per year owning a computer (Common Sense Media, 2011). In addition, only 10% of lower-income parents have a smart device (e.g. iPhone) versus 34% of upper-income. Only 2% of lower income families have a tablet computer versus 17% of upper-income families, and 38% of lower-income parents do not know what an “app” is, compared with 3% of higher-income parents, 47% of whom have downloaded apps for their children, compared to only 14% of the lower income parents (Common Sense Media, 2011).

Although children in North America grow up in an environment full of digital devices, this fact does not necessarily mean children understand their

digital worlds. Research indicates that simply providing access to technology is not enough to convey an understanding of technology; it is also important to understand the social context of the use of technology (Zillien & Hargittai, 2009; Palfrey & Gasser, 2008). Thus, schools likely play a key role for narrowing the gap and providing more equal opportunities for STEM education and learning about innovative new technologies.

Recently, some researchers have moved from emphasizing the digital divide to discussing the participation gap (Gee, 2013b; Jenkins, 2009a; Jenkins, 2009b). A combination of cheaper, faster, and better technology tools has led to more widespread access to technology, especially through community organizations, such as libraries and afterschool programs. Therefore, the problem may no longer be that children do not have access at all: instead, the problem may be the quality and quantity of the access to technological tools and the differences in the ways in which children engage in using digital technologies, referred to as the participation gap.

For example, lower-income and ethnic minority students are less likely to have parental guidance when using the Internet (Gutnick, et. al., 2010). Children from higher income households were more than twice as likely as children from the lowest income households to use a computer to complete homework assignments (Roberts & Foehr, 2008) and more likely to spend time on high-quality websites that develop school-based skills or are more reading-intensive (Gutnick, et. al., 2010). Five to nine year old African American children spend more than an hour per day, on average, playing video games, compared to 54

minutes for Hispanic five to nine year olds and 41 minutes for Caucasian five to nine year olds (Gutnick, et. al., 2010).

On average, across groups, children spend almost as much time with digital technologies as they do learning in school, leading to an opportunity to leverage these technologies (Shuler, 2007), but the quality of the exposure to digital devices and ways of using the tools are varied. Thus, we should ask ourselves not only what kinds of technologies young people are using, how often, and in what context, but also, what are young children doing with these technologies? Are the children using the technologies in a way that is consistent with developmentally appropriate practice? Are the technologies supporting the children to engage in the developmental tasks appropriated for their age? How can we leverage these technological tools to teach universal and transferable skills?

In sum, new digital technology use appears to be ubiquitous and growing, especially among young children. However, the overall trend does not mean every child has equal access to new technologies at home or at school, a key point to be mindful of when the term “digital native” has been a common part of the lexicon around digital learning.

Coding Tools for Young Children

This section provides an overview of programming tools for young children in order to highlight the influence of these tools on education, in general. Further elaboration on ScratchJr and self-regulation, specifically, appear in the next chapter.

Both use of general computer applications and use with early computer programming languages have shown positive impacts on cognitive abilities such as abstraction, problem solving, and structural knowledge (Haugland, 1992; Clements & Sarama, 2002; Wang & Ching, 2003). Learning to code in early childhood has also shown a positive impact on sequencing skills (Kazakoff & Bers, 2012; Kazakoff, Sullivan, & Bers, 2013). Bers and colleagues' work has also explored how young children can understand powerful computational ideas, such as control flow, loops, branches, and sequencing in addition to developing sophisticated levels of computational thinking (Bers, 2008; Bers, 2010a; Bers & Horn, 2010; Bers et. al., 2002).

Computer programming or coding is the basis of digital technologies and an important skill for 21st century literacy (Rushkoff, 2010; Jenkins, 2006). Twenty-first century literacy is defined as the ability to use technological tools, collaboratively solve problems, share information globally, manage, analyze, critique and synthesize information and multi-media, and behave ethically in complex environments (NCTE, 2008). Computer programming, with developmentally appropriate programming languages is one area of particular interest in K-12 education (Bers, 2008; Resnick, et. al., 2009; <http://csta.acm.org>). At the core of the idea for a developmentally appropriate programming language is one that gives children the ability to understand the notion of coding as a sequence of steps or instructions designed to direct an object's behavior (Bers, 2008; Bers, Flannery, Kazakoff, & Sullivan, in press; Kazakoff & Bers 2012; Kazakoff, Sullivan, & Bers, 2013).

Past research has shown technologies which enable children to become programmers can be powerful tools for teaching children both old and new skills and concepts (Bers, 2008; Bers, Flannery, Kazakoff, & Sullivan, in press; Bers & Horn, 2010; Cejka, Rogers & Portsmore, 2006; Papert, 1980). The largest body of research on young children and computer programming has likely been conducted with LOGO, a text-based programming language developed by Seymour Papert (1980). When introduced to LOGO in a supportive classroom environment, computer programming has been found to impact a wide range of cognitive skills in early childhood, including computational thinking (Clements, 1999; Liao & Bright, 1991), meta-cognition (Clements, 1986; Miller & Emilhovich, 1986) and transfer of skills in problem representation, problem solving, and debugging (trouble-shooting) to activities outside of the programming context (Klahr & Carver, 1988; Salomon & Perkins, 1987; Degelman, Free, Scarlato, Blackburn, & Golden, 1986).

Counterclaims to LOGO suggested that even if children learned to write sequences of code, they still may not have the ability to understand other aspects of computing, for example, how to use a mouse, which may cause unnecessary cognitive distress (Elkind, 1986). Moreover, young children also lack the metacognitive and self-evaluative language to fully express their emotions surrounding digital technology use (Fleer, 1999). However, a lot has changed since the introduction of LOGO. Even though the version of ScratchJr introduced herein was introduced on a laptop with a mouse interface, the ScratchJr team realizes that touch screen interfaces a likely more developmentally appropriate for

the fine motor skills of young children and the first public release version of ScratchJr will be on tablets. Concerns also persist around technology use taking over physical activity (Bus & Newman, 2009) and computer use being an isolating experience for children (Cordes & Miller, 2000). At least in early childhood, children are still allowed recess and the break in their day for physical activity. Furthermore, the proposal in this study is that digital technology is used as a tool in the classroom and not to replace an out-of-classroom activity. In regard to student isolation using computers, both this study (see Case Study chapter) and prior studies (Druin, 1998; Muller & Perlmutter, 1985; New & Cochran, 2007; Wartella & Jennings, 2000) have demonstrated that children actually interact more frequently around computers than during other classroom activities.

New research and theories based on innovative programming environments support the argument that children's programming of animations, graphical models, games, and robots with age-appropriate materials allow children to learn and apply core computational thinking concepts such as abstraction, automation, analysis, decomposition, modularization, and iterative design (e.g., Lee, et al., 2011; Mioduser, Levy & Talis, 2009; Mioduser & Levy 2010; Resnick, 2006; Resnick et al., 2009). Children who used LOGO in kindergarten were also found to have sustained attention, self-direction, and they took pleasure in discovery (Clements, 1987). A large-scale study of children using the LOGO programming language, (Clements, Battista, & Sarama, 2001) demonstrated that children in Grades K – 6 scored significantly higher on tests of

mathematics, reasoning, and problem-solving. The researcher's explanation for the difference in scores is that when children engage in coding activities, and thereby create a sequence of commands for the computer to read, the children externalize their inner thought process. This externalization of inner thoughts may make the child's thought process more available for reflection and understanding (Clements, Battista, & Sarama, 2001). This reflection and understanding of one's thought process is also known as metacognition, which is a component of EF (Garon, Bryson, & Smith, 2008). Perhaps then, the metacognitive aspects of learning to code could also contribute to developing EF skills in children who learn coding skills in early childhood.

Young children who used LOGO also developed higher order thinking skills and transferred their knowledge to map reading, interpreting the rotation of objects, and demonstrated understanding of a wide variety of logic and math knowledge (Clements 2002). Furthermore, computer programming has been found to positively impact creativity and emotional response in children with learning difficulties (Clements & Swaminathan, 1995). A review of early work in the field also suggests children who participated in programming tasks typically scored around sixteen points higher on various cognitive-ability assessments compared to children who had not participated in such programming (Liao & Bright, 1991). Finally, research indicates new technologies, such as cell phone apps, may have a positive impact on vocabulary and literacy skills in three to seven year old children (Chiong & Shuler, 2010).

The invention of the tablet-based computer lent itself to an influx of coding and creation tools for young children in the form of “apps” (mobile applications). Some “apps” have aspects of basic programming skills, but rely heavily on words, such as Daisy the Dinosaur (<http://www.gethopscotch.com/>), while others mask programming actions, such as ToonTastic (<http://launchpadtoys.com/toontastic/>). In the physical coding space, there are BeeBots (<http://www.terrapinlogo.com/bee-botmain.php>) and Cubelets (<http://www.modrobotics.com/cubelets/>); however, BeeBots have very simple commands and Cubelets mask the actual programs. These tools seem to engage young children playfully with technology but appear to obscure some of the important aspects of creative coding, such as the notion of creating a visible sequence of commands to impact an (physical or graphical) object’s actions. When designing tools to teach programming skills to young children, it may be beneficial to make the individual code clear – and include high levels of complexity – so that the children may be more actively engaged with debugging, problem-solving, metacognitive thinking, and sequencing which could potentially translate to other areas of learning.

For the more complete programming tools, there is Lego® WeDo (<http://www.legoeducation.us/eng/categories/products/elementary/lego-education-wedo>), a robotics and programming kit for ages seven and up; Kodu (<http://fuse.microsoft.com/projects/kodu>), a graphical programming language for ages eight and up; and Scratch (<http://scratch.mit.edu/>), a graphical programming language for ages eight and up. While these tools are quite robust (e.g., they

contain hundreds of coding blocks), they are not necessarily appropriate for use in early childhood in their standard forms (e.g., the blocks are based primarily on words and math concepts young children cannot understand). The following subsection will describe one of these tools in further detail – Scratch – as Scratch is the starting point for ScratchJr, the software used in this study.

Scratch. Scratch is a free and widely used graphical programming language designed to make sophisticated programming accessible to children as young as eight. (Resnick, et. al., 2009). With Scratch, users can program interactive (meaning children are actively engaged with, and manipulating, digital media rather than passively viewing) art, stories, animations, simulations, and games by snapping together sequences of graphical blocks that represent instructions specified by a label written on each block (Resnick, et. al., 2009). There are additional tools in Scratch, such as a paint-editor, graphics, and audio editing features. In using Scratch, children learn literacy and math concepts in addition to problem-solving skills (Resnick, 2007).

Scratch is not intended for children as young as five. The set of programming instructions is vast and is labeled entirely with text. This poses challenges for pre or early readers to navigate and keep track of the options available (Flannery, et. al., 2013). To program characters to travel around the Scratch screen, users need an understanding of the Cartesian coordinate system, positive and negative numbers, and a sense of the relative size of numbers up to at least +/- 500. (Resnick, et. al., 2009, Flannery, et. al., 2013). This knowledge is appropriate for much older children and adults and impedes young children's

attempts to use Scratch. However, Scratch has been a powerful and popular tool for introducing older children, teens, and adults to creative computation and programming (Resnick, et. al., 2009).

Returning to the concept of computational thinking, addressed in the introductory section, Brennan and Resnick (2012), created a definition of computational thinking involving three specific dimensions that apply to Scratch (and, in turn, may also apply to ScratchJr). They are computational concepts, computational practices, and computational perspectives. Computational concepts refers to areas such as sequencing of programming instructions, parallel programming by either giving multiple programs to one character or giving two or more characters programs to act out together, and programming events, such as “start on green flag” (Brennan & Resnick, 2012).

The second component of Brennan and Resnick’s definition of computational thinking is computational practices, including iterative design, testing and debugging, and abstracting and modularizing. The final component is computational perspectives, which is when children make connections between the technological tool and the real world, by asking questions and making connections with others (2002).

This more specific definition of computational thinking, specifically related to Scratch/ScratchJr, is mentioned to highlight the connections between computational thinking, math, literacy, and problem solving. Sequencing is related to early math and literacy skills (Brown & French, 1976; Kamps, et. al., 2008; Kazakoff & Bers, 2012; Kazakoff, Sullivan & Bers, 2013; Paris & Paris,

2003), while iterative design, testing, and debugging are related to problem-solving. Recalling that math, literacy, and problem solving have all been correlated with self-regulation in prior studies (Espy, et. al., 2004; McClelland, et. al., 2007; Zelazo, Carter, Reznick, & Frye, 1997), it is possible that activities that build computational thinking, such as learning to program with Scratch or ScratchJr, may also contribute to the development of other skills, such as math, literacy, problem-solving, and self-regulation. Of course, there is still much work to be done in these areas to demonstrate concrete, measurable connections.

In sum, digital technologies, such as Scratch, ScratchJr, and others, have the potential to make learning more social, collaborative, and networked (Gee, 2010; Jenkins, 2006) and, perhaps, contribute to other areas of the curriculum as well. Since early research on computer technologies, researchers have found that, when children work at a computer, they speak twice as many words per minute than when engaged in other non-technology related play activities such as play dough and building blocks (New & Cochran, 2007); and, they speak to their peers nine times more than when working on traditional puzzles (Muller & Perlmutter, 1985). Children, when working on computers, are also more likely to ask other children for advice and help, even if an adult is present, thus increasing child-child socialization (Wartella & Jennings, 2000). Even in situations where each child has an individual computer or their own piece of digital equipment to work with, children still choose to form groups (Druin, 1998). There is an image that comes to mind of children sitting in silence, programming their own individual computers or tablets. While the software and hardware can be individualize, there

appears to be – from this current ScratchJr study and from those studies cited above – something about working with digital technology that makes the classroom environments less formal and more interactive, meaning, the children will get up from their seats, talk to one another, collaborate, and seek help from their peers over their teachers even when working on individual projects.

When one thinks of emerging technologies, particularly computing, cognitively heavy, syntactically complex tasks come to mind. When we think of the use of new technologies in early childhood, and the developmental level of children ages two to seven, the more prevalent issues are those of how can we use technology to enhance learning, particularly, how do we create technologies for children who are just beginning to develop self-regulation skills? The next chapter details an overview of the ScratchJr software, curriculum, and addresses the design considerations taken into account in ScratchJr, specifically addressing early childhood self-regulation skills.

Chapter Three: Developing ScratchJr

Taking into consideration the developmental needs highlighted in Chapter Two, the ScratchJr team set out to design and evaluate an engaging and developmentally appropriate graphical programming language for young children from five to seven years of age. This chapter describes the ScratchJr software, related curriculum, and connections to self-regulation in greater depth.

Designing ScratchJr: Software

ScratchJr, the technological tool at the center of this study, is a graphical programming language for young children (target ages of five through seven). ScratchJr was created based on the graphical programming language for older children, Scratch (<http://scratch.mit.edu>), which was designed for children ages eight and older, that has been used by a wide following of children, adolescents, and adults. The ongoing development of ScratchJr is a collaboration between the DevTech research group at Tufts University, MIT Media Lab's Lifelong Kindergarten group, and the Playful Invention Company of Montreal (PICO), with funding from the National Science Foundation (NSF) (DRL-1118664). At the core of the ScratchJr project is the idea that children in kindergarten through grade two have very limited options as compared to older children when it comes to powerful educational technologies that specifically account for their developmental levels. This is especially true in terms of fine motor control, reading level, and self-regulation (Flannery, et. al., 2013).

During the initial phase of the ScratchJr project, the ScratchJr team partnered with teachers, school administrators, and young children to create a

developmentally appropriate tool, which met the needs of early childhood educators and targeted specific learning outcomes. The ScratchJr project focused on learning outcomes in the following areas:

- Discipline-specific knowledge: ScratchJr targets national and state early childhood educational frameworks for math and literacy and can be used to create projects across topic domains.
- Foundational knowledge: ScratchJr aims to engage children in building foundational knowledge across skill sets such as sequencing, causality, symbol understanding, pattern recognition, estimation, and prediction (visual-spatial integration).
- Problem-solving skills: In the process of creating interactive digital projects with ScratchJr, children engage in activities that apply the scientific method/engineering design cycle (ties into self-regulation).

The final version of ScratchJr will aim to include three key components: a developmentally appropriate interface, with both touch screen and mouse interface options; an embedded library of curricular modules that meet federal and state standards for early childhood education; and an on-line community and resource library for early childhood educators and parents.

The version of the ScratchJr software used in this study is the second testing version. Results from this study, and subsequent studies, informed the design of the eventual marketplace version of ScratchJr. The second testing version is comprised of a main project editor that contains tools for selecting and

drawing characters and settings (New characters, text, and settings can be added by clicking large buttons labeled with icons representing each component. Multi-scene stories may be created in sequences of up to four pages (thumbnails of which appear on the right-hand side.) The set of characters included on each page is managed from the list to the left of the story page.

[Insert Figure 1] At the center of the project editor is the story page, which is the scene under construction. New characters, text, and settings can be added by clicking large buttons labeled with icons representing each component. Multi-scene stories may be created in sequences of up to four pages (thumbnails of which appear on the right-hand side.) The set of characters included on each page is managed from the list to the left of the story page.

[Insert Figure 1. about here]

[Insert Figure 2. about here]

The list of possible programming blocks categories lies along the center of the editor. Children display one block category at a time by clicking on the color-coded selectors. Dragging blocks from the palette into the scripting area below activates the programming block. The blocks can be snapped together to create programs that are read and played from left to right. Figure 3 presents an example of the “motion blocks” category.

[Insert Figure 3. about here]

The grid (shown in Figure 4) overlaid on the story page provides a concrete, countable unit of measurement for a character’s actions. It can be

toggled on and off as needed (e.g., programming vs. presenting a project). The “Flag” and “Stop” buttons start and interrupt the programmed animation.

[Insert Figure 4. about here]

Designing ScratchJr: Curriculum

The ScratchJr software continues to be iteratively developed and tested for both home and school use. The intention is to create both a developmentally appropriate programming tool and an accompanying supportive curriculum. During the iteration of ScratchJr described in this dissertation, the ScratchJr curriculum was in a very early form, having undergone changes from initial pilot tests.

Research suggests curricula that are designed to improve self-regulation skills in addition to teaching early academic abilities may be most effective for school success (Blair & Razza, 2007). To note, the ScratchJr curriculum used in this study was a pilot version and a somewhat hybrid approach to curriculum design. Part of the ScratchJr project as a whole (although not part of this dissertation work) included testing various curricula with different levels of scaffolding (For an example of a ScratchJr card used as part of the curriculum, see Appendix 1.) The children who took part in this version of the study used the most open-end, least-scaffolded version of the curriculum, which included one day of open-ended exploration, five structured lessons, and two days of semi-structured projects. At the conclusion of the phase of the ScratchJr project described in this dissertation, a new and significantly more structured curriculum was introduced (see <http://www.scratchjr.org/teach.html> for an example).

For primary analysis, this study considered the eight ScratchJr lessons that were taught in both Classroom Pre16 and Classroom Pre8. The eight lessons included in the Fall 2012 analysis for both classrooms were:

1. Introduction/Free Explore: After a brief overview of the interface of the ScratchJr software, children were encouraged to explore the tool on their own and discuss what they had discovered with the researchers, their teachers, and each other.
2. Airplane Fly Across USA¹ (**Airplane Lesson**): This lesson introduced the “start on flag” block, motion blocks, and the method for changing number parameters. Children programmed an airplane to fly across a map of the USA.
3. Character Race (**Race Lesson**): In this lesson, students programmed three characters to race against each other at varying speeds. This lesson introduced the set speed block and reinforced the motion and “start on flag” blocks (see Figure 5 below).

[Insert Figure 5. about here]

4. Characters Dance (**Dance Lesson**): The dance lesson introduced the concepts of multiple scripts on one character and sound blocks. Children programmed one character to provide both the music and dance steps, while they programmed a second character to provide just dance steps. This lesson also introduced the “start on contact” block, which initiates a program when two

characters bump into each other and the “repeat forever” block, which plays a programming script continuously.

5. Sunset (**Sunset Lesson**): Children programmed a sunset in this lesson. The show, hide, and go home (reset) blocks were introduced. Start blocks and motion blocks were reviewed.

Children that finished the sunset lesson early had the opportunity to attempt a “moon rise” lesson.

6. Characters Greet Each Other (**Greet Lesson**): In this lesson, children learned how to program characters to have a conversation when they bump into each other. This lesson reinforced the “start on contact” start block and introduced speech bubble blocks, “send a message”, and “start on message received” blocks.

7. & 8. “About Me” Project (**Project Lesson**): Children were instructed to make four ScratchJr pages about themselves, first brainstorming with paper and crayons. The four pages were (1) a picture of themselves with their name, (2) a page about school, (3) a page about home, and (4) a page about what the student wanted to be when he/she grew up. This task proved to be too intense for a two-day project. Final projects were considered successful if two of these four pages were created and something was programmed. Children received a lesson on how to use the “add a page” feature before beginning their projects.

Design Features of ScratchJr: Self-Regulation

When learning to code with a new programming language, such as ScratchJr described above, children may enter the process with varying levels of self-regulation, which may, in turn, have an effect on their progress throughout the coding activities. While children from low-income and ELL families tend to enter school with lower levels of self-regulation, studies have found that the interactions between self-regulation and academic achievement are not related to SES or ELL status (McClelland & Wanless, 2012). Meaning, whether or not a child is from an ELL or low-SES background, if that child has or develops a high level of self-regulation, which will likely translate to better learning outcomes.

ScratchJr was created specifically for five to seven year old children, taking into account their varying levels of self-regulation (as well as motor skills, reading levels, etc.). The following section describes self-regulation (working memory, attention, inhibition control) specific design considerations taken into account when ScratchJr was developed.

ScratchJr contains approximately 30 programming blocks in contrast to the 100-plus blocks in the original Scratch. Fewer programming blocks were included in ScratchJr in order for the program to be less taxing on a child's working memory. In addition, ScratchJr contains a compound programming block (e.g., Hop) with the possibility of including more compound blocks in future designs. A "hop" is comprised of a "forward" block plus an "up" block. This feature allowed a child to create a meaningful series of steps in a program

with a minimal number of blocks to monitor, thus allowing for complex programs to be created with less strain on working memory.

There are limited “quick fix” buttons in ScratchJr. Most actions must be programmed (e.g., increasing a character’s size is done with programming blocks rather than with a + or – button). This design is intended to assist children who are working on attention and inhibition control skills by reducing the temptation to click repeatedly for an immediate reward.

The number entry in ScratchJr is limited to the scope of the screen: 15 units vertical, 20 units horizontal – which corresponds to children’s understanding of number sense and accounts for variations in inhibition control when using the keypad. This restriction was implemented due to children inputting excessively long numbers in pilot studies, e.g., “2062476460.” There is also a graph with numbers that overlays the story page area. This feature allows the children to count the spaces in which their character moved or needs to move, potentially allowing a break from mental math for those students who do not yet have strong working memories.

Although many other design factors were considered when developing ScratchJr, this dissertation focused on self-regulation. The key design choices listed above were implemented in ScratchJr in order to lessen strain on young children’s developing self-regulation skills. As such, a study was designed to assess the relationship between self-regulation and learning to code with ScratchJr. The study design is detailed in the next chapter.

Chapter Four: Method

This chapter describes the methodology of the ScratchJr study.

Participants are described, and the specific tools, assessments, and data collection measures used in the two kindergarten classrooms that participated in this dissertation work are outlined.

Design-Based Research

The ScratchJr project followed a Design Based Research (DBR) methodology, which means that, the project was iteratively refined in the process of its implementation. By definition, “design experiments are extended (iterative), interventionist (innovative and design-based), and theory-oriented enterprises whose ‘theories’ do real work in practical educational contexts (Cobb, Confrey, diSessa, Lehrer, Schauble, 2003, p. 13).” With DBR studies, understanding can be at the classroom learning ecology level, rather than the larger theoretical level (Cobb, et. al., 2003). DBR focuses heavily on an interventionist approach to research, where researchers are involved in sustained educational interventions lasting weeks, months, or years (Bell, 2004). For the ScratchJr study data were collected in classrooms during the regular school day. The interventionist, DBR approach allowed for real-time, iterative changes to the curriculum design and ScratchJr software to understand a classroom of learners, in depth over several months as they learned to code; despite all the unintended variables that came with this approach (e.g., snow/hurricane days, attendance issues, technical difficulties).

Five different kindergarten to second grade (K-2) classrooms across two

schools participated in the ScratchJr study during the period of October 2012 to May 2013. This dissertation study focuses on two of those kindergarten classrooms during a three-month period, beginning with the student's first exposure to the ScratchJr programming language. The initial curriculum design sought to strike a balance between scaffolding targeted use of software components with the desire to observe how and what the students produced with the software from their own imaginations. Hence, there was opportunity for free exploration combined with structured curriculum challenges and a semi-structured final project. All of these components of the curriculum were works in progress.

During the instruction sessions, the researchers acted as teacher-experimenters (Cobb, et. al, 2003). Researchers, research assistants, and classroom teachers adapted curriculum in the moment to work around technical challenges and classroom dynamics. The curriculum also underwent a major revision between the Fall 2012 and Spring 2013 semesters, based on the Fall 2012 data. This study focuses on Fall 2012 only.

Sample

The sample was a volunteer, convenience sample. Participants in this study were members of two kindergarten classrooms located in a densely populated suburb of Boston, Massachusetts with a range of economic and ethnic diversity. Participating teachers and parents/guardians of students provided signed informed consent prior to participation.

The initial sample size for this study was 39 students: 19 in Classroom Pre16 (labeled as such because the students took a self-regulation pre-test

followed by 16 ScratchJr lessons) and 20 students in Classroom Pre8 (labeled as such because they took a self-regulation pre-test followed by 8 ScratchJr lessons). In Classroom Pre16, one child left the school during the beginning of the study, another child refused to participate in pre-testing due to unfamiliarity with researchers, and another could not be used for any video analysis. The average age at the start of the study was 5.51 years old (SD = 0.30). Classroom Pre16 was 57.9% female and classroom Pre8 was 50% female. The classrooms did not vary significantly in terms of age, sex, or pre-testing scores.

Demographic data are reported at the school, rather than individual student level. School-level demographics reports were used because of (1) a history of low rate of return of consent forms (and concern that making the forms longer with background surveys would have delayed the process further) and (2) the multiple translations required for the consent forms. It was decided to make the consent forms as simple to complete as possible in order to ensure all students returned the forms and were allowed to fully participate in the study. This decision meant leaving off additional questions the researchers would have liked to ask regarding demographics, home technology use, etc.

For the 2012-2013 school year (the study year), the participating school reported in this study was composed of 40.1% Hispanic students, 37.1% White, 14.6% African American, 6.7% Asian, and 1.5% Multi-Race/Non-Hispanic students. About 75% of the school was classified as “high needs,” including students with a first language other than English (41.2% of the total student population), low-income students (68.4% of students), students receiving free or

reduced lunch (68.2% of students), and students classified as Special Education (24.7% of students) (<http://profiles.does.mass.edu>).

Although data were not collected on individual children's first language, some inference can be made regarding the dominant language spoken at home based on the consent forms returned. In this study sample, all consent forms returned from classroom Pre16 were in English. Some children in classroom Pre16 could speak a language other than English at home as evidenced by that fact that the school classified 41.2% of the population at the as English Language Learners (ELL). However, ELL parent(s)/guardian(s) chose to return the English-language consent. In classroom Pre8, four students returned consent forms in languages other than English: two in Spanish and two in Portuguese. These children represent 20% of the Pre8 sample and 10.5% of the final 38 student sample. Again, these four children could be an underrepresentation of the ELL students in the classroom if parent(s)/guardian(s) chose to return the consent form in English even though English is not their primary language.² However, knowing which participants turned in consent forms in languages other than English allows for some further data analysis of these four students.

Recruitment. The participating classrooms were chosen for several reasons: first, the school's location within the community surrounding the university, second, the prior research relationship between the Developmental Technologies (DevTech) Research Group and the kindergarten teachers, as well as the relationship between the elementary school and the university, and third, the opportunity to deliver new educational technology tools to an "at-risk" school.

The school district in this study was classified for 2012 – 2013 as Level Three, meaning “districts with one or more schools among the lowest-performing 20% based on quantitative indicators; schools scoring in the lowest 20 percent statewide of schools serving common grade levels, regardless of NCLB accountability status (<http://profiles.doe.mass.edu/help/data.aspx>).”

In the spring prior to the start of this study, kindergarten parents had the option to attend a kindergarten information day. The parents who attended the kindergarten information day were informed about the ScratchJr study in the spring prior to the study’s start. This informational session provided parents an opportunity to ask the researchers questions with language translators present and gave the parents several months to email the researchers any questions. All children in the kindergarten classrooms participated in the ScratchJr study regardless of whether or not their parents attended the kindergarten information day. The information day was a chance for the school to promote the collaboration between the elementary school and university while easing parent concerns and/or increasing excitement around the project.

In the fall of 2012, a description of the ScratchJr study and informed consent documents, including documents translated into Portuguese and Spanish, were sent home with the children. All children returned consent forms by the time their classrooms participated in the hands-on computer work. A third classroom had difficulty with consent return rates during the initial phase of the study (from September through December 2012). The entire third classroom returned their consent forms in time to participate in a future ScratchJr

curriculum, which occurred in February and March of 2013. The third classroom's ScratchJr and pre/post activities are not included in this dissertation but are mentioned here for ethical reasons to establish that all three classrooms in this school did eventually participate in the ScratchJr intervention.

Procedure

In the fall of 2012, after one year of iterative design and pilot testing, the second version of the ScratchJr software, designed for laptop computers, and the accompanying curriculum was pilot tested. During the 2012-2013 school year, two schools and five classrooms participated in the study. This dissertation focuses on two of the kindergarten classrooms in one of the schools, a local, public elementary school.

The primary researcher worked with kindergarten teachers and school administration to set up all technology in the classrooms, including a secure wireless router that would be attached during lessons to the classroom's Internet and 23 laptops that would be brought into the classrooms each day.

For the research reported here, pre-testing using the Head-Toes-Knees-Shoulders (HTKS) assessment was conducted during the week prior to the start of the study in the two participating classrooms. During the study, one or two research assistants and the primary researcher rotated between classrooms, four days per week over the course of approximately 75 days. While the students were at recess, the researcher and assistant(s) set up the laptops with the ScratchJr software and screen capture software (Camtasia). When the children returned from recess, the researcher led a group lesson where she introduced a targeted

concept in ScratchJr, projecting the software on a large screen and walking the children through the lesson's goal. The children then attempted to complete the same challenge in ScratchJr on their individual laptops. At the conclusion of the session (approximately 45 minutes), the researcher and assistant(s) saved the ScratchJr programs and Camtasia videos (see page 74 for more on fidelity to ScratchJr program curriculum).

The two classrooms each participated in the study two days per week on average. One classroom received sixteen lessons of ScratchJr. The other classroom received a combination of eight lessons of ScratchJr and eight lessons of KidPix (www.kidpix.com), a digital creation software that allows for computer programming/coding, following the same teaching format. The classrooms rotated participation days (one classroom participated Mondays and Thursdays and the other participated Tuesdays and Fridays). At the conclusion of all the lessons, both groups were administered the HTKS post-test assessment.

Data were collected from several sources from October through December of 2012. Data sources included direct assessments of children using the Head-Toes-Knees-Shoulders assessment (HTKS), an inventory from teachers using the Behavior Rating Inventory of Executive Function Form (BRIEF), and video/audio recordings using saved projects and Camtasia audio/video/screen recordings. Approximately 75 days passed between pre-test and post-test.

A mixed-methods approach was applied to this dissertation work. The children's individual trajectories through their programming experiences are rich sources of qualitative data, while standardized assessments (BRIEF, HTKS)

quantify children's levels of self-regulation to index any changes in self-regulation between the beginning and end of the ScratchJr learning to code intervention.

Several different types of data were collected from these sessions including logs of children's projects, videos of children using the software, and HTKS pre-test and post-test data. Children participated in structured lessons and semi-structured projects during the coding sessions. Data on the five structured lessons and a semi-structured project that both classrooms participated in are analyzed in the following chapters using quantitative and qualitative analysis with illustrative case studies. The individual lessons are described in the next section.

[Insert Figure 6. about here]

Lesson Structure. The first eight lessons of ScratchJr include one introductory free explore lesson, five structured lessons, and two open-ended project sessions taught by a ScratchJr researcher and assisted by one or two research assistants. Video data were collected on all children whose parents consented. The ScratchJr lessons occurred during choice/enrichment time (when children choose to work on areas of their individual interests) in the kindergarten classroom for approximately 45 minutes, twice per week. Classroom Pre16 received 16 lessons of ScratchJr while Classroom Pre8 received eight lessons of an alternative, developmentally appropriate technology tool, KidPix (www.kidpix.com) that allowed for digital creative self-expression without coding followed by eight lessons of ScratchJr.

Four days per week were spent in the classroom, alternating between Pre16 and Pre8. The researcher plus one or two assistants would set up the laptops with ScratchJr and Camtasia (or KidPix) while the students were at recess. Upon returning from recess, the students would have circle time during which the researcher would introduce the new lesson, which focused on targeted programming blocks. For example, the Airplane Lesson focused on “Start on Green Flag,” “Move Forward,” “Change # Parameter,” and “End”. The initial instruction was demonstrated with a projector and large screen. After the lesson, students went to their individual laptops to try the lesson for themselves. The researcher, research assistants, and classroom teachers would assist with scaffolded prompts but were guided not to give direct assistance except for technical difficulties. For example, a researcher could guide a child through asking what category of blocks he or she needed in order to make a character move if he or she seemed to be stuck and did not know what to do next. Eventually, some interface items (e.g., deleting a character) did prove to be more troublesome for teachers and students alike and were more explicitly explained. At the end of the class, students were told to save their projects. The second through fifth lesson and one project lesson are included in analysis. The initial exploration lesson was provided as an orientation to the software but not as a means for evaluation. The project lesson was evaluated on the last day of the project, whenever possible.

Measures

HTKS

McClelland and colleagues developed the Head-Toes-Knees-Shoulders (HTKS) self-regulation assessment (McClelland, et. al., 2007; McClelland, et. al., 2009, Ponitz, et. al., 2008, Ponitz, McClelland, Matthews, & Morrison, 2009). The HTKS was developed as a tool specifically for classroom settings in that it is quick to administer (less than 10 minutes per child) and is designed for children in this study's age range. The HTKS version used for this study is the HTKS – extended, which includes 30 test trials, for possible score of 0 to 60. For each test trial, a child could score two points if he or she responded correctly, one point if he or she self-corrected, and a zero points if he or she responded incorrectly.

The HTKS comes in two forms, A and B. No significant differences were found between forms (Ponitz, McClelland, Matthews, & Morrison, 2009). For this research, the HTKS was administered as Form A for the pre-test and Form B for the post-test to minimize familiarity with one set of exercises over another. The pre- and post- tests were administered to all classroom groups.

Research assistants learned to administer the HTKS by reading the protocol, watching the training video, and practicing the protocol. The HTKS was administered in a structured way. First, the child was asked to follow commands as instructed. For example, the researcher or assistant stated, “touch your head,” and the child then touched his or her head. Next, the child was instructed to respond contrary to the actual instruction. For example, the administrator stated, “touch your knees” and child's correct response was to touch

his or her shoulders, or the administrator stated “touch your toes,” and the child’s correct response was to touch his or her head. If the child responded correctly, he or she was awarded two points. Incorrect responses earned zero points and self-corrected answers (e.g., if the child moved towards the wrong body part but ended with the correct body part) earned one point. Research assistants administered the HTKS individually to kindergarten participants at the school, in the hallway, for approximately 10 minutes per child for pre-test and post-test.

Inter-rater reliability, scoring agreement, and test-retest reliability are high for the HTKS, which has alphas of .93 (Ponitz, 2008; Ponitz et al., 2009; McClelland, et al., 2007; Wanless, McClelland, Tominey, & Acock, 2011). Scores on the HTKS are significantly correlated with teacher ratings of children’s self-regulation in preschool (McClelland, et. al., 2007) and with teacher ratings of self-regulation and parent ratings of attention and inhibitory control in kindergarten (Ponitz et al., 2009).

BRIEF

Teachers of students in Classrooms Pre16 and Pre8 completed a Behavior Rating Inventory of Executive Function (BRIEF) form for each student (Gioia, Isquith, Guy, Kenworthy, 2000). Teachers were given the BRIEF assessment in order to correlate their observations of children’s self-regulatory behavior inside of the classroom with researchers’ observations of children’s self-regulation during the HTKS assessments. The BRIEF measures executive function (EF) across seven clinical scales (initiate, working memory, plan/organize, monitor, inhibit, shift, emotional control) (Gioia, Isquith, Guy, Kenworthy, 2000).

Teachers assign a 1 if children never engage in a behavior, 2 for sometimes, 3 for often. Teachers were provided a guide sheet from the researcher, verbal instructions from the researcher, and contact information if they had questions. In addition, instructions were included on the BRIEF form. Teachers spent approximately 10 minutes per child completing an inventory scale for children in both classrooms. The teachers completed these forms during winter break. Both the researcher and an undergraduate researcher assistant calculated raw scores in order to check for transfer, calculation and addition errors.

The BRIEF is a standardized, valid, and reliable measure. Normative data were based on approximately 1,400 parents and 700 teachers. There was high internal consistency ($\alpha = .80-.98$) and test-retest reliability ($r_s = .88$ for teachers) (Gioia, Isquith, Guy, Kenworthy, 2000).

Camtasia Videos of ScratchJr. Children's computer screens of ScratchJr projects, along with faces and audio while using ScratchJr, were recorded using Camtasia software (see Figure 7) (<http://www.techsmith.com/camtasia.html>). Researchers coded the videos two to three times, in several areas. First, they completed a checklist of software features and coding blocks (see Appendix 2). Second, they recorded how much time the child spent on-task (defined as working towards the lesson assigned to the child) versus off-task (defined as working on an unrelated project in ScratchJr or not attending to ScratchJr). Third, they indicated if a child completed the goal of the day's lessons.

[Insert Figure 7. about here.]

The checklist of software features was developed based on listing all the

components found within the ScratchJr software interface. The on/off-task rubric was developed based on the curriculum or project assigned for the specific lesson being coded. The checklist scoring is based on marking when the child “attempts” and “achieves success” with a particular programming block. Inter-rater reliability was calculated twice. First, inter-rater reliability was coded by assigning all ten potential raters the same four videos to code – one randomly selected Airplane Lesson, one randomly selected Race Lesson, one randomly selected Greet Lesson, and one randomly selected Project. Krippendorff’s Alpha was used to calculate inter-rater reliability across these four videos and ten raters. Krippendorff’s alpha is considered a standard for reliability calculation in content analysis and similar fields when there are multiple human observers (Hayes & Krippendorff, 2007). For the videos examined, there were ten coders, with $\alpha = .832$, meaning a generally acceptable level of reliability (0.7 or higher is usually considered acceptable, especially for a piloted coding scheme) (Freelon, 2013).

A second sample of inter-rater reliability was assessed throughout the video coding process. Randomly selected videos were put back into the coding pool for a second rater to rescore. If there were any discrepancies on these videos, the primary researcher also viewed the videos and weighed in on the discrepancy. When in doubt, the lower score prevailed (e.g., a child would be assigned a score of attempted rather than understood a block if understanding was not clear so as to not potentially overstate the success of the students on the new software tool). In this sample, there were eight raters across thirteen videos with

two or three raters, plus the primary researcher rating each video. Krippendorff's alpha for almost all combinations was good: $\alpha_{abc}=0.855$, $\alpha_{bcd}=0.900$, $\alpha_{cbg}=0.850$, $\alpha_{cvg}=0.873$, $\alpha_{cdg}=0.932$, $\alpha_{acdh} = .868$, $\alpha_{acf} = .952$, although for one video, $\alpha_{ace}=0.593$. Video data were reviewed by the primary researcher and corrected where needed. In total, 17 videos, comprising 9.4% of the video sample ($n = 181$ videos) used in this dissertation were inter-rated. Additional videos were inter-rated but are not included here as they were for lessons not included in this work (e.g., Free Explore time and Spring 2013 lessons).

Computed Measures

The following three variables: Programming Score, Goal Complete Score, and Time on Task Score, were calculated by Camtasia video analysis.

Programming score. Programming score was calculated from the interface checklists, based on watching the videos of children working with ScratchJr. Programming score is intended to be a measure of what targeted blocks and interface elements of the ScratchJr Lesson the child attempted and understood. Coders watched videos of the participants coding with ScratchJr and completed checklists of all possible coding blocks and interface elements (see Appendix 2 for a copy of the checklist). Coders marked each block or interface element as follows: blank if the child did not use the block or interface element, a “•” if the child attempted to use the block or interface element and a “✓” over the dot plus documentation of the timestamp of the video if the child demonstrated understanding of the block or interface element. If there was any doubt about understanding, the child was left with a “•” meaning “attempted.”

Demonstrations of understanding included verbal expressions of the block's purpose or targeting a block for an intended action, such as "I need this block [selects hide block] to disappear a cat" or, if the child selected an incorrect block for the program, but then debugged the program correctly. For example, if a child placed a green flag start block, a move left block, and end block on her airplane, and then she clicked the green flag start block and watched her program, she would receive credit for understanding the green flag starts her program and that the red end blocks ends her program. If she looked quizzically at the screen and then replaced her move left block with a move right block, and replayed the action before declaring "I'm done," she received credit for understanding both the left motion block and the right motion block because she knew what the left motion block did, knew it was wrong, and replaced it with the right (correct) motion block for the assignment.

Programming scores were assigned in several steps. First, the blanks, dots, and checks on the checklists were translated into scores of 0, 1, or 2, respectively. Next, the scores were split between which blocks and interface elements were targeted for the assigned lesson and which blocks and interface elements were not. This step was done to try to control for the use case in which a child would get "attempt" points for just dragging every block into the scripting areas. Final programming scores were assigned on a 0 to 5 scale, according to the following general guidelines:

0 = Did not attempt any coding blocks.

1 = Attempted primarily non-targeted coding blocks (the child did not earn 1s or 2s for targeted blocks) but did receive 1s or 2s for other coding block(s).

2 = Attempted but did not understand any targeted blocks (no 2s for targeted) and/or understood two or more non-targeted blocks (2s or greater).

3 = Attempted and understood some of the targeted blocks (At least one target category is a 2).

4 = Understood majority of targeted blocks (e.g., 75% of targeted categories are 2s).

5 = Understood all targeted blocks (All 2s in targeted categories).

Goal Completion Score. Goal completion scores assessed if the child worked toward and completed the day's assigned ScratchJr lesson. The child received a score of 0 if he/she did not attempt the lesson, a score of 1 if the child attempted the lesson, and a score of 2 if the child completed the lesson.

Definitions of attempt and complete varied by lesson. Lessons were completed if the child met the goal of the lesson. Lessons were considered attempted if the child made an effort to work towards the goal. Items listed under a score of "1" for each lesson are descriptions of actions that describe the child worked towards the lesson goal even if he/she did not achieve the goal. The children were told that after they completed the assigned lesson, they could work on their own projects. The hope was that this was motivating for students who did not want to work on the class assignment. In addition, all lessons allowed for personalization (e.g., selecting/coloring characters). However, there were instances when children did not work towards the lesson goal. Sometimes these projects were complex and interesting. This complex issue in scoring is discussed further in the Case Study Chapter. The Goal Complete scoring guidelines are as follows:

Airplane

Score	Description
2	Airplane Character + Start Block + Motion Block + End + Changed # Parameter and/or used multiple motions
1	Airplane Character + Start Block + Motion Block

0	Anything Else
---	---------------

Race

Score	Description
2	Three characters with Start Block/Speed Block/Motion Block on each
1	Any script on more than one character AND/OR one character programmed with speed block AND/OR all three characters plus an attempt to program at least one of them
0	Anything Else

Sun

Score	Description
2	The successful program needs to “show” when the program starts running, “hide” when the program starts running and move in between (either up/down or left/right). The program can be either a sun or a moon. The child needs to attempt to run the program (to negate just a copy/paste of the ScratchJr card).
1	Use of show or hide and a motion block. AND/OR Use of go home block and a motion block. AND/OR Use of show or hide block and go home block.
0	Anything Else.

Dance

Score	Description
2	One character must have motions and sounds. A second character must have motions. These characters must interact on a bump start block.
1	Use of sounds. AND/OR Use of bump with two characters. AND/OR Motion scripts on two or more characters.
0	Anything Else

Greet

Score	Description
2	Send-Receive message combination is work with speech bubbles and child tested his or her program.
1	Multiple scripts including an envelope and/or dialogue on at least one character. AND/OR Envelops on two or more characters. AND/OR Dialogue on two or more characters. AND/OR Script on one character that includes start-message-envelope.
0	Anything Else.

Project

Score	Description
2	At least one page with a functional program clearly (by program or by child statement) representing self, home, school, future (“when I grow up”).

1	Worked on coding something (not just coloring or adding characters) but it was not related to self/home/school/future
0	No programming blocks used.

Time on Task Score. Time on Task Score is an analysis of whether or not children were focused on the ScratchJr Lesson, another area of the ScratchJr interface (e.g., the paint editor), or wandering the classroom. The Time on Task assessment is by no means a perfect assessment. Nonetheless, since attention span and staying on task towards goals is predictive of future success and is related to self-regulation (McClelland, Acock, Piccinin, Rhea, & Stallings, 2013), the Time on Task variable was an important one to include in this preliminary analysis. The way of assessing Time on Task is not without flaws but gives a general overview of how the children spent their time while using the ScratchJr software. To assess time on task, videos were watched and coded for on-task behaviors versus off-task behaviors and time stamps were noted.

Primarily two people coded videos. All potential coders watched and coded the same four videos for training before coding any additional videos. After training multiple coders, only two coders coded the videos because of budget and time constraints, not because of inter-coder agreement issues. Throughout the Time on Task video coding process, random videos were assigned to both coders to code. Due to the fact that coders were manually pausing and writing timestamps from the Camtasia software for behaviors that changed sometimes by the second, some freedom was given for exact matches in time. The primary researcher entered all the data into a spreadsheet, calculated the percentage of time on and off-task based on the coders' timestamps and reviewed

the coders' notes. Any major discrepancies were watched, verified, and recoded, if needed, by the primary researcher. Since videos varied in length of time due to lesson variation and technical issues, percentage of time on-task was used as the unit of measurement for Time on Task Score.

In sum, in this chapter, the study participants and methodology were presented. Two standardized measures were highlighted: the HTKS self-regulation assessment delivered by the research team and the BRIEF executive function inventory, completed by teachers and designed to be used as a method of triangulation. In addition, three variables that were researcher-created, based on the video data collected in the classroom, were described: Programming Score, Goal Completion Score, and Time on Task Score. In the next chapter, results of participants' work with ScratchJr, focused on self-regulation, will be analyzed through four key variables: HTKS, Programming Score, Goal Completion Score, and Time on Task Score.

Chapter Five: Results

Classroom Pre8 consisted of 20 students who remained for the duration of the study, and classroom Pre16 consisted of 18 students who remained for the duration of the study. The two classrooms did not vary at the start of the ScratchJr study in terms of sex, $t(36) = .33, p < .74$, age, $t(36) = .82, p < .42$, or HTKS pre-test score, $t(33) = .27, p < .79$. BRIEF correlations were not calculated at the start of the study because the BRIEF assessment required teachers to have known students for at least three months and data collection on self-regulation began less than one-month into the school year.

Thirty-five children completed the HTKS self-regulation pre-test ($M = 24.91, SD = 15.51$). Thirty-eight children completed the HTKS post-test ($M = 28.37, SD = 17.32$). As would be expected, for the total sample, HTKS pre-test scores are correlated with HTKS post-test scores ($N = 35, r(33) = .69, p < .01, 95\% CI [0.46, 0.86]$). HTKS pre-test scores had a cluster of eight students at the floor with a score of 0. However, this distribution of pre-test scores is in the normal range for skewness, but trending towards that floor effect (skewness = $-.517$). See below.

[Insert Figure 8. about here.]

This sample has 21% of the sample at the floor with a score of zero. This is possible due to the demographic makeup of the sample. While individual demographic data on the students was not collected so it is impossible to draw any direct conclusions or correlations, the school itself has a low-SES and high ELL status population. In prior studies of self-regulation with high-ELL and low-SES

learners, there have been correlations with low self-regulation scores, meaning, a child with low-SES or with ELL status may be more likely to have a score of zero on the HTKS assessment. Thus, since this study was conducted in a school with both a ELL and low-SES students, the sample has a greater tendency than other samples to cluster at the floor.

BRIEF Scores

The BRIEF executive function assessment was intended to be used as a triangulation method for the HTKS assessment. The two classroom teachers scored the BRIEF while the primary researcher and research assistants scored the HTKS. The HTKS was a moment in time while BRIEF was designed to assess the prior three months. The BRIEF reported global and subscale accounts of the child's EF compared to standardized sample of peers. Global BRIEF scores were correlated with class assignment ($N = 38$), $r(36) = .83$, $p < .01$, 95% CI [0.69, 0.91], which is considered a strong correlation. However, the BRIEF Global score did not correlate with either the HTKS pre-test score or HTKS post-test score. Subscales that might appear to be related to the HTKS, such as Inhibition Scale, and Working Memory Scale, were also not correlated with either the HTKS pre-test or HTKS post-test score. Results are summarized in Table 1 and Table 2.

[Insert Table 1. about here.]

[Insert Table 2. about here.]

Global BRIEF scores were correlated with class assignment ($N = 38$), $r(36) = .83$, $p < .01$, 95% CI [0.69, 0.91], which is considered a strong correlation. Class Assignment ($N = 38$) was correlated with each sub-scale, Inhibition ($r(36) = .53$, $p < .01$, 95% CI [.25, .73], Working Memory ($r(36) = .77$, $p < .01$, 95% CI

[0.59, 0.87]), Emotional Control ($r(36) = .71, p < .01, 95\% \text{ CI } [.50, .84]$), and Planning/Organization ($r(36) = .87, p < .01, 95\% \text{ CI } [.76, .93]$). No correlations were found between the BRIEF and the three programming variables: Programming Score, Time on Task, and Goal Completion Score. Those results are summarized in Table 3, Table 4, and Table 5.

[Insert Table 3. about here.]

[Insert Table 4. about here.]

[Insert Table 5. about here.]

Fidelity of ScratchJr Program Implementation

Before diving into the research questions, it is important to consider whether the children actually participated in the research study. That is, did the children engage in the ScratchJr programming activities the researchers created and interact with the software the ScratchJr team developed?

One way to answer this question is to reference the Goal Complete variable. Across five assigned lessons (Airplane, Dance, Race, Sun, Greet) and a final project, there were a total of 181 videos coded. Across these videos, 85% of the children attempted the assigned ScratchJr projects; thus, children attempted to work towards the assigned ScratchJr goals a majority of the time. About 34% percent of the ScratchJr Lessons and projects were successfully completed. This score on project completion leaves substantial room for curriculum development. As this was the first time running this curriculum at full-scale, successful completion of the curriculum was not the primary of a focus. The main goal was to actually engage students in the act of learning to code, and this was achieved because all children attempted at least one lesson, and most children attempted

most lessons. Even for the children who were not working toward the assigned goal (15% of instances), there was never an instance when a child flat out refused to engage with the ScratchJr software. Not working towards a goal primarily meant the child was coloring in the paint editor rather than coding. In addition, the average Programming Score was ($M = 2.95$, $SD = .79$) out of a possible 5. ScratchJr is designed to have a low floor and high ceiling, for use for children from kindergarten through second grade. It was not expected that kindergarten students would be experts or understand all elements of the ScratchJr interface. That kindergarten students were attempting most blocks, and understanding some, is considered a success.

Research Question One

Question One asked to what extent do differences in initial levels of self-regulation, measured by the HTKS assessment, account for differences in ScratchJr coding skills in kindergarten. Since classrooms Pre16 and Pre8 do not vary in sex, age, or HTKS pre-test score, the two groups were combined for the analysis of Programming Score, Goal Completion Score, and Time on Task score within the five structured ScratchJr lessons (Airplane, Race, Dance, Sunset, Greet) and semi-structured ScratchJr project compared to the HTKS pre-test score as a baseline level of self-regulation (page 49 presents a description of the lessons).

In order to assess if baseline level of self-regulation, as measured by the HTKS assessment, was correlated with coding skills in ScratchJr, analyses were run with three different variables: Programming Score, Goal Completion Score,

and Time on Task Score. These three variables were chosen to attempt to give a complete picture of the participants' time spent with the ScratchJr interface, meaning, his or her understanding of the ScratchJr programming blocks and interface components, the use of the programming blocks and interface elements towards a provided goal (during the lessons) and semi-structured goal (during the project), and a way to account for time spent on task coding and off-task with other activities during the ScratchJr time.

Programming scores and self-regulation. The Programming Score is a representation of the ScratchJr programming blocks and interface elements attempted and understood by the child. (Refer to page 66 for description of the Programming Score variable).

Programming scores for the Airplane, Race, Sun, Dance, and Greet Lessons were compared to baseline self-regulation as measured by HTKS pre-test scores. Projects were not included as there were no targeted blocks for use in analysis as there were with the other lessons.

Results indicated there were no correlations between self-regulation as measured by the HTKS pre-test assessment and any of the Programming Scores, either by individual lesson or an average score across lessons. Three individual lessons were correlated with one another, indicating some consistency in children's approaches to ScratchJr programming. The Airplane and Race lessons were correlated ($n = 23$, $r(21) = .52$, $p = .01$, 95% CI [.14, .77]), as well as the Race and Dance lessons ($n = 21$, $r(19) = .55$, $p = .01$, 95% CI [.15, .79]). Results

for individual lessons and an overall average compared to self-regulation are summarized in Table 6.

[Insert Table 6. about here]

No significant correlations were found for Programming Score and Age or Programming Score and Sex.

Goal completion scores and self-regulation. The Goal Completion Score is a representation of whether or not the child completed or attempted to complete the assigned ScratchJr Lesson. All ScratchJr Lessons and the Project were evaluated for this variable (page 68 presents a description of the Goal Completion variable and scoring rubric).

Goal Completion scores for the Airplane, Race, Sun, Dance, Greet and Project Lessons were compared to baseline self-regulation as measured by HTKS pre-test scores. Children appeared to approach coding in ScratchJr consistently across lessons, in regard to completing the assigned lesson, as four Goal Completion Scores correlated with each other across lessons, Race-Dance ($n = 21$, $r(19) = .57$, $p = .01$, 95% CI [.18, .80]) and Sun-Greet ($n = 23$, $r(21) = .45$, $p = .03$, 95% CI [.05, .73]).

There was a moderate correlation between the average Goal Completion Score across all lessons and the project, and, initial level of self-regulation as measured by the HTKS pre-test ($n = 34$, $r(32) = .36$, $p = .03$, 95% CI [.03, .63]), meaning children with higher levels of self-regulation had higher rates of goal completion. For individual lessons, results indicated a moderate correlation between initial level of self-regulation as measured by the HTKS pre-test score

and Goal Completion Scores for the Sun lesson ($n = 29$, $r(27) = .47$, $p = .01$, 95% CI [.12, .71]). Goal Completion Score was also moderately correlated with Age, both overall, $n = 37$, $r(35) = .35$, $p = .03$, 95% CI [.03, .61] and for the Sun Lesson, $n = 31$, $r(29) = .36$, $p < .05$, 95% CI [.00, .63]. No other individual lessons, nor sex, were significantly correlated with self-regulation as measured by the HTKS pre-test assessment. Results are summarized in Table 7 below.

[Insert Table 7. about here.]

Time on task scores and self-regulation. A sub-sample of lessons were coded for Time on Task Score. The Airplane, Race, and Project Lessons were coded. The percentage of time spent on task was correlated with initial level of self-regulation as measured by the HTKS pre-test assessment (see page 70 for description of the Time on Task Score Variable). No significant correlations were found in the Time on Task analysis. The results are summarized in Table 8.

[Insert Table 8. about here.]

The average Time on Task percentage was ($M = 43.2\%$, $SD = 16.7\%$). However, Time on Task peaked for the Race Lesson ($M = 59.7\%$, $SD = 23.3\%$), while the Airplane ($M = 33.5\%$, $SD = 23.3\%$) and Project Lessons ($M = 38.8\%$, $SD = 28.1\%$) had closer averages. The graph in Figure 9 presents the trajectory of Time on Task among the Airplane, Race, and Project Lessons.

[Insert Figure 9. about here]

No correlations were found for Time on Task Score and Age or Time on Task Score and Sex.

Outlier Analysis

Due to the lack of significant results for Question One, further exploratory analysis was conducted with particular attention paid to potential outliers that could be skewing the small data sample. In the spirit of factor analysis, for which this study does not have enough data to perform, it was decided to examine scatterplots with best fit lines, see which points loaded on the fit line, and then compare the data points above the line to the data points below the line to see if there was a difference between those two outlier groups.

Scatterplots of Programming Scores, Goal Completion Scores, and Time on Task Scores (x2) were shown to three independent raters. The primary researcher also participated in the exercise. A prompt was given: “Some points are clearly on the line. For example, case number 28. Some points are clearly not on the line, for example, case number 7. Can you circle the other points that you consider ‘close enough’ to the line to be clustered around the line? Alternatively, you may cross out points you judge to be clearly off the line.”

A point was considered in agreement to be an outlier and used if 100% of the four coders agreed on the point (32% of points). If three of the four coders agreed, the 95% confidence interval lines were consulted in SPSS. If the point also fell outside of these lines, it was included (68% of points).

Programming score analysis. When graphically comparing average Programming Score to HTKS, coders agreed there did not seem to be any outliers. Therefore, no above the line/below the line comparisons could be run. However, examining the scatterplot below, there is a notable cluster of children vertically around zero. These are the children who scored around a zero or two on the

HTKS pre-test of self-regulation and scored anywhere from 1 to 4 points (on a 0 – 5 point scale) for a programming score. This pattern persists across the graph. Children clustered at 20, 30, and the mid-40s have similar ranges in scores. This visually depicts why there was no relationship between self-regulation as measured by the HTKS and Programming Score – the children had a range of Programming Scores regardless of their initial HTKS pre-test. Some children with low levels of initial self-regulation had high average programming scores, such as case number 34, and some children with high initial levels of self-regulation had on the lower end of average programming scores, such as case number 5. This distribution of scores at the HTKS floor speaks to the diversity in Programming Scores regardless of initial self-regulation level. Children with low levels of self-regulation were able to program had high levels of achievement while others did not perform as well. More on specific cases can be found in the case study chapter. This diversity in Programming Score happened even at the higher levels of self-regulation, as well.

[Insert Figure 10. about here]

Goal completion score analysis. A Goal Completion Scatterplot was completed by including all individual lessons and the project lesson. Again, there is the floor effect of 0-2 on HTKS Pre-Test with a range of scores on the Goal Completion variable. However, in this scatterplot, the points look more tightly clustered around the fit line. The outlier agreed upon by the coders for above the line is number 37 and below the line are numbers 7, 15, 26, 27, and 33. HTKS pre-test scores of the above the line group of outliers were compared to the below

the line group of outliers using an independent-samples *t*-test and no significant differences were found, $t(4) = .92$, $p < .41$, 95% CI [-28.1, 55.7]; $d = .92$ between the groups in HTKS pre-test scores.

Due to the small number of data points in this analysis, it is difficult to draw any conclusions. However, the graph in Figure 11 suggests a linear relationship between Average Goal Completion Score and HTKS pre-test score when ignoring these outliers. This relation makes sense, as a small correlation was found between the HTKS pre-test and Goal Completion Score. Note that the outliers that fell outside the circle, but were not included in the outlier analysis, are the ones for which the coders did not reach 75% agreement (e.g., 34, 36).

[Insert Figure 11. about here]

Results indicated a small correlation between initial self-regulation level as measured by the HTKS and overall Goal Completion Score ($N = 34$, $r(32) = .36$, $p = .03$, 95% CI [.03, .63]), as well as a small correlation between initial self-regulation as measured by the HTKS and one individual lesson (Sun) ($N = 29$, $r(27) = .47$, $p = .01$, 95% CI [.12, .71]). The Sun and Greet Lessons have a small correlation with each other ($N = 23$, $r(21) = .45$, $p = .03$, 95% CI [.05, .73]). No other individual lessons are correlated with self-regulation as measured by the HTKS. Visually, when considering Figure 11, there does appear to be a linear relationship between Goal Completion Score and HTKS.

Time on task score analysis. Time on Task Score graphs were evaluated in two parts. First, they were evaluated using two lessons (Airplane, Race) plus the Project Lesson and then they were evaluated using just the Project Lesson.

With the Goal Completion and Time on Task variables there was a hypothesis that children may be more focused on, or spend more time on, a project that was more personally relevant. Therefore, the partially open-end project was also evaluated independently.

The outliers agreed upon by the coders for above the line were 7, 25, 32 and for below the line were 27. Due to the small number of outliers, a *t*-test is not being reported for this group. The graph is presented in Figure 12. Again, there are large variations in Time on Task percentages for children who scored at the floor level on the HTKS pre-test assessment. Children who scored very high on the HTKS assessment, 40 points or above, primarily scored above average for Time on Task as well, with the exception of Data Point 27 (Child 31). He will be further discussed in the case study chapter.

[Insert Figure 12. about here.]

Time on Task analysis of the project alone is a bit more revealing. The outliers agreed upon by the coders for above the line were 3, 7, 14, 19, 22, 32 and for below the line were 12, 15, 21, 23, 27, 28 (see Figure 13). HTKS pre-test scores of the above the line group of outliers were compared to the below the line group of outliers using an independent-samples *t*-test and a significant difference was found, $t(11) = 1.145$, $p < .04$, 95% CI [-8.4, 26.4]; $d = .69$ between the two groups' HTKS pre-test scores. These results suggest that there is a relationship between self-regulation score as measured by HTKS for the group of students identified as outliers above the fit line ($M = 25.29$, $SD = 18.01$) and self-

regulation score of children identified as outliers below the fit line ($M = 34.33$, $SD = 7.20$).

[Insert Figure 13. about here.]

Results across the three variables, based on means, indicate that children numerically (in terms of Programming Score ($M = 3.63$, $SD = 1.38$), Time on Task Score ($M = .597$, $SD = .23$), and Outcome Score ($M = 1.45$, $SD = 0.63$)) were more successful at lessons where they were, anecdotally, more excited and enthusiastic (for example, continually trying to go back to their Race programs weeks later and finding the Greet program to be incredibly silly/funny). This potentially has implications for curriculum development.

Despite Airplane being designed as the first and simplest lesson, Race ($M = 3.63$, $SD = 1.38$), Sun ($M = 3.13$, $SD = 1.26$), and Dance ($M = 2.92$, $SD = .98$) all had higher average Programming Scores than the Airplane Lesson ($M = 2.62$, $SD = 1.27$). Race was the most successful lesson, in terms of Programming Score average ($M = 3.63$, $SD = 1.38$), Outcome Score ($M = 1.45$, $SD = 0.63$), and Time on Task ($M = .597$, $SD = .23$). For comparison, overall averages were of Programming Score ($M = 2.95$, $SD = .79$), Outcome Score ($M = 1.19$, $SD = 0.37$), and Time on Task ($M = .432$, $SD = .16$).

Research Question Two

Question Two asked to what extent, if any, did learning to code with ScratchJr impact self-regulation skills? In order to explore this question, self-regulation scores were assessed in the Pre8 and Pre16 classrooms at roughly the same time during the pre-test and post-test but the Pre16 classroom was exposed

to eight more sessions of ScratchJr than the Pre8 group during the time period in between the self-regulation pre-test and post-test (approximately a 75 day span). To control for technology exposure and the novelty of technology use in kindergarten, an alternative technological tool for kindergarten, KidPix, was used in the Pre8 classroom when they were not using ScratchJr. KidPix allows for graphical animations but with no exposure to coding.

Pre-test self-regulation scores as measured by the HTKS were compared to post-test scores as measured by the HTKS using a between-within ANOVA with classroom as the between variable and HTKS as the within variable. No significant main effect was found, $F(1, 33) = 2.23, p = .14, n^2_{\text{partial}} = .06, 95\% \text{ CI} [-1.2, 8.0]$.

Despite the lack of a significant finding, there does appear to be some relationship between HTKS pre-test and post-test scores and participating in additional lessons of ScratchJr, based upon gains in HTKS score. On a 60-point scale, there was a 3.98 point average gain in HTKS score across both classrooms between pre-test and post-test. However, for the Pre16 class ($N = 16$) there was a 5.92 point increase between the HTKS pre-test ($M = 25.69, SD = 14.97$) and post-test ($M = 31.61, SD = 15.28$), a gain of 23.0% compared to a 1.19 point increase in Pre8 ($N = 19$), a 4.9% increase ($M_{\text{pre}} = 24.26, SD_{\text{pre}} = 16.34; M_{\text{post}} = 25.45, SD_{\text{post}} = 18.89$).

In sum, the data analysis for Questions 1 and 2 was performed on a pilot sample of 38 children. Although no major conclusions can be drawn, and all results should be interpreted with caution, there are some trends to highlight from

these results and discuss further. First, there does not appear to be a relationship between Programming Score and initial level of self-regulation as measured by the HTKS pre-test assessment. Second, there does not appear to be a direct relationship between Goal Completion Score or Time on Task Score and self-regulation as measured by the HTKS assessment when interpreting correlations between these variables. However, additional inspection of the graphical data suggested a possible link between the more open-end and motivating tasks and self-regulation as measured by the HTKS pre-test. Third, although the relationship is not significant, there does appear to be some benefit to self-regulation scores for the group who learned to code with ScratchJr for sixteen lessons compared to the group who learned to code for eight lessons. Finally, the BRIEF assessment used to triangulate the HTKS assessment did not correlate with HTKS scores. However, a correlation was found between BRIEF scores and class assignment.

A primary reason why these results need to be interpreted with caution is because of the small sample. One of the benefits of a small sample, however, is the ability to dig deeper into some of the reasons why data outcomes may or may not be what are expected. The next chapter highlights some of the outlier cases and describes some more typical approaches to ScratchJr programming lessons.

Chapter Six: Case Studies

This chapter highlights several individual examples of kindergarteners in two classrooms that learned to code with ScratchJr. The individual case studies provide an overview of some varied approaches to the ScratchJr software. They also deliver a more detailed exploration of some of the children's approaches to learning to code and probe into some unexpected findings from the prior Results Chapter. This chapter also introduces some of the limitations of the ScratchJr program as well as new areas of research focus that will be further discussed in the subsequent chapter. *Note: All names have been changed.*

Overview Examples

High Achievement. Child 12. Classroom Pre16.

Overall, across all variables: Programming Score, Goal Completion Score, and Time on Task Score, one child consistently stood out among the highest scorers, Child 12. This case study is Child 12's approach to ScratchJr. Peter is 5.41 years old. His HTKS pre-test score is 42/60. His average programming score is 4.33/5, average Goal Completion score is 1.75/2, and his average Time on Task is 77%.

When examining Peter's programs and videos further, what stood out was that in addition to consistently completing the projects assigned, he went above and beyond the assignments. For example, with the Race project, the child finished the challenge early and then asked about repeats (repeats had not yet been taught). He then practiced making repeating programs for a cat. With the Sun project, he programed the sun project, moved on to an extension that was

provided if there was extra time (Moon Project), and then extended the project further when he made the sun blink and changed the speed by which the sun moved down the screen (the sun sets).

Perhaps it should not have come as a surprise, then, that his father informed the researchers during a demo day that he had been teaching Peter regular Scratch for the better part of a year. This example is illustrative of both a high level of programming achievement and the need to collect baseline information on students' use of software and digital devices at home. This case study also speaks to "preparatory privilege" where there is a tendency to assume children who come from homes with more technology are more talented in the field or have more interest. This misconception comes from the child having more exposure and practice, yet, can still cloud a teacher's perception. Children and teachers may think a child is not interested or not able to engage in computing work; however, the child may be fully capable and, instead, not have been exposed to the tools in an engaging way (Margolis, Estrella, Goode, Holme, & Nao, 2008).

Low Achievement. Child 25. Classroom Pre8.

Overall, across all variables: Programming Score, Goal Completion Score, and Time on Task Score, one child consistently stood out among the lowest scorers, Child 25. This is Child 25's approach to ScratchJr.

Jack was 5.52 years old. His HTKS pre-test score was 0/60. His average programming score was 1.67/5, average Goal Completion Score was 0.40/2, and Time on Task average was 15.6%. Of course, every child has individual strengths

and weaknesses when approaching ScratchJr. Language may have been a factor in Jack's performance as his consent form was returned in Spanish. However, this is just one of several factors that could have been at play in Jack's performance. It is worth noting a language other than English is spoken in the home, however, is not possible to draw a direct connection between the consent form language and Jack's primary language.

During the Airplane Lesson, he primarily colored and experimented with the "add text" feature (see Figure 14). There are no data from the Race Lesson as the video is too blurry. During the Sun Lesson, Jake was out of his seat and not at his computer despite constant redirection by the classroom teaching assistant. No video data exist for the Dance Lesson. There was a change in pattern, however, during the Greet Lesson. He receives full Goal Completion credit for this lesson (2/2) and his Programming Score for this lesson was his highest overall (3/5). (Note: It is possible to achieve a goal but not receive full credit for programming since a child can use all the correct blocks and run the program, thus completing the goal, without fully understanding what the blocks mean. This scoring strategy is one way of mediating excess outside help from teachers and research assistants.) For the Greet Lesson, the classroom teacher can be seen in the video walking him through each and every step. For the Project, Jack receives one point of credit toward goal completion because he talks about adding his sibling, however, he spends most of the time coloring a baby character, in outer space, during the time he is at his computer.

This case study demonstrates several points. First, this child, who was otherwise unable to complete a program or even attempt to use the programming blocks, successfully completed one of the harder ScratchJr lessons when continually prompted by a teacher. Perhaps for this child, and other children who scored a zero on self-regulation, classroom dynamics, temperament, and motivation are the limiting or moderating factors when learning to code with ScratchJr rather than the interface. Furthermore, he is a child encompassed by the floor effect in the HTKS pre-test scores. As noted, his consent was returned in Spanish and prior research suggests students with ELL status tend to enter school with lower levels of SES. This could be one of many reasons for the low score. To note, all HTKS assessments in the classroom were delivered in English.

Even when the teacher was sitting with Jack and prompting him through the Greet Lesson, he still needed to stand up and wander around periodically. This may speak to Jack's temperament and sustained attention. Second, this child may have more at play than just developing self-regulation skills. His background was not disclosed to the researchers; however, accommodations were made throughout the ScratchJr lessons in order to ensure he was only sat next to specific children where he would be least likely to be distracted or distract others. Finally, Jack was able to build code within the ScratchJr interface when he had a teacher prompting him; an activity, which he was not able to do when there was less scaffolding. This example also raises the question for future curriculum design: how much prompting and scaffolding is too much that it causes the child

to be no longer engaged in the problem-solving and debugging that is fundamental to learning to code?

[Insert Figure 14. about here.]

Consistent Coding. Child 18. Classroom Pre16.

This next case study presents a child whose Programming Score (3.33/5) and Time on Task percentage (56.6%) were above average and she was at the top of the group for Goal Completion Score (2/3). Sarah is an example of a consistent, successful coder in ScratchJr. Sarah was 5.65 years old. She scored 36/60 on the HTKS pre-test assessment, which is in the high range for this sample.

To highlight the Race Lesson in particular, Sarah completed the lesson, but her final ScratchJr program for this lesson does not appear to demonstrate that she completed the targeted goal (three characters racing with variable speed). Instead, it looked like this:

[Insert Figure 15. about here.]

Final versions of ScratchJr lesson programs saved online looking unlike they did in the videos is fairly typical of most students and a reason why the Camtasia videos were so helpful in data analysis. Children had a tendency to log into programs where they had already been or seek out favorite prior programs, thus overriding the work they had previously completed.

After completing the Race Lesson goal around minute 25 of class, Sarah then began coloring her caterpillar character from brown and yellow to pink and blue. After that, she colored her chicken a bright red and her pig a bright pink.

After re-running her race a few times with the brightly colored characters, she deleted the pig and chicken, added a girl in a blue dress, replaced a field background with a classroom background, and added the additional girl, bunny, and adult characters (as seen in Figure 15). She then saved this project. On a second page, she made a new race project, on a nighttime field background, with the brightly colored pig, caterpillar, and chicken characters she had created (see Figure 16). As a final step, she changed the night field back to a daytime field. However, the final state of the Race program that was saved is just one particular classroom scene (see Figure 15). During later lessons of ScratchJr, Sarah went back and deleted the second page with the completed (for a second time) Race program.

A child overwriting their prior projects was a common occurrence in this version of ScratchJr. The above example is typical of the differences seen between programs saved online and the achievements captured on the videos. Furthermore, this transition of moving back and forth between programming a character and coloring the character was also common throughout the ScratchJr programming process. The point in having a robust paint editor was that creativity and self-expression are important in early childhood classrooms. Making characters more personally meaningful also encourages children to more deeply engage with the ScratchJr software. However, in order to progress through the ScratchJr curriculum, a balance needed to be struck between spending all of one's time coloring versus coloring just enough to tailor the characters to the specific lesson. For the average child, like Sarah, the balance worked out and

children both engaged with the ScratchJr coding lessons and made the lessons personally relevant and meaningful by adding their creative, artistic, and personal flair.

[Insert Figure 16. about here.]

Result Call-Outs

The following case studies are more in-depth discussions of particular outliers and data points beyond what was reported when interpreting the results. This section sheds some light on what may have been happening beyond just the numbers.

Case 7. Child 2. Classroom Pre16.

Jane appears as a high scorer for Time on Task score for both overall Time on Task (70.0%) and for the Project 63.8%). However, this is a case when Time on Task does not necessarily mean that Jane was attending to programming or working towards a programming goal. Jane frequently colored, but, she frequently colored within the theme of the lesson. In cases like this, it looked like Jane was working towards the lesson goal and was “on task”. However, in most cases, she did not add programming blocks or waited until the end of the lesson to try to code and did not finish. Furthermore, being less vocal at soliciting the attention of limited adult resources in the classroom, she also spent more time waiting for help than her peers. For example, in the Airplane Lesson, she waited for help with deleting a miscellaneous character before she attempted to program her airplane character and thus ran out of time. Yet, waiting for help counts as still

being “on task” in the coding scheme as children should not be faulted for technical glitches, lack of adult assistance, needing questions answered, etc.

This case is included to demonstrate some additional considerations for future studies. These considerations are, namely, digging deeper into scaffolding needs and assistance from adults as well as differentiating between off-task coloring, on-task coloring, and on-task coding. For most children, coloring was clearly off-task, like coloring cats in space (see Figure 17). It is also worth considering in future studies if “off-task” for activities like coloring is a negative. It could be that the children who are engaged in cognitive heavy tasks need the coloring breaks to process what they are learning. Somewhat relatedly, this case study highlights the need for differentiated curriculum that appeals to different temperaments and styles of learners. Perhaps a curriculum that more specifically integrates the artistic “breaks” that are more cohesive within the curricular theme.

[Insert Figure 17. about here.]

Time on Task Project Outlier.

Recall that for Time on Task score related to the Project, children who scored high on the HTKS assessment, 40 points or above, primarily scored above average for Time on Task score, with the exception of data point 27 (Child 31). Upon further analysis, it appears that Adam spent the class time coloring multiple cats black with red eyes. His HTKS pre-test score was a 41, his average Programming Score was a 3.25/5, average Goal Completion Score was a 1.0/2 and average Time on Task is 23.6%. While Adam did usually code during other lessons, for the Project video, it seems he colored cats instead. Unfortunately,

there is no further explanation as to why. His overall Time on Task is lower than the average for the class (43.2%). Of course, multiple factors could be at play, including the child's individual temperament (although his behavior on this lesson was not consistent with prior behavior) and outside influencing factors from the home or classroom environment. It could also be that the personal project was not an engaging enough curricular lessons compared to the prior lessons.

Data Collection and Analysis Case Studies

The following case studies illustrate specific points about use of ScratchJr and ScratchJr data collection.

Andy's Cars. Child 8. Pre16 Classroom.

Andy worked consistently on the assigned lessons until it came time for the Project Lesson. Then, he took the open-ended nature of the project and to the extreme. He can be heard saying "I can do ANYTHING I want" on the video for the first day of the Project Lesson. Doing "anything" was a misunderstanding of the semi-structured project instructions. However, Andy took his understanding of being able to do "anything" and created an unexpected and complex ScratchJr project.

Unfortunately, there are no video data for the Airplane or Race lessons for this child. The saved projects reveal an Airplane Lesson with his name but without programming and a Race Lesson with the correct program for the pig but not for any other character. These examples demonstrate that saved project logs are not greatly useful for data analysis as they only save the final state of the project.

There are video data for Andy's Dance, Greet, and, Project lessons. For the Dance and Greet lessons Andy follows the assigned task, working towards the assigned goals. He expresses excitement "awesome!" during the Greet Lesson when he programs the cat and dog to interact with one another.

For the Project Lesson, Andy does not follow the class assignment of "make one page about you, one page about home, one page about school, and one page about what you want to be when you grow up." Instead, he makes a scene of a town with multiple cars driving. This program is interesting for its complexity in the area of parallel programming, or putting multiple programming scripts on multiple characters to run simultaneously. A screen shot of the program is below. This case represents the interesting dilemma in this research – what happens when the child is technically not doing the assignment but working towards something really complex? In this case, Andy did get partial points because coding anything during the Project Lesson phase counted towards an attempt at doing a Project. However, should this complex program be awarded fewer points than a program that has just one character, yet fit with the theme? An attempt to negate the complexity issue was made by requiring at least two pages of the Project program to be coded for the Project program to count as complete. (Anything beyond two pages was too much to require of the kindergartners in the time allotted). However, this case points to several areas of consideration in future studies: how to account for innovative, sophisticated programming that is not technically assigned; how to maintain better logs of the

children's programs when videos fail; and how to determine a feasible assignment length for the kindergarten curriculum versus first or second grade curriculum.

[Insert Figure 18. about here.]

“Blinking Cat” Vignette. Child 21. Classroom Pre8.

The first exposure lessons were not coded for this study, but are potential material for future study, especially in the area of motivation and how children first approach a novel digital tool. (Note: As this case study was used as a vignette in a forthcoming edited volume [Kazakoff, in press], it is included here in vignette style.)

Violet moves her mouse cursor around the screen to click on the paintbrush icon next to the orange cat. A window opens with a selection of colors. She smiles as she spots pink (her favorite) and clicks first on the color, then the paint bucket icon, and again on her cat. She giggles as her cat turns pink and then she returns to the main ScratchJr programming screen. Violet then browses the menu of programming blocks and selects a “green flag” for her cat followed by a “hide” icon and a “show” icon. She snaps these three blocks together and then clicks the “green flag” and watches what happens. Her cat disappears from view and reappears. She smiles, excitedly claps her hands together, and asks classmates to come see her program. Violet then systematically adds alternating “hide” and “show” icon blocks until she has a program almost as long as the screen. Every time she adds two more blocks, she plays the program by clicking the “green flag” and mouths quietly “invisible,

visible, invisible, visible” in time with the cat’s performed actions. She continues to excitedly request that the rest of her classmates come view her work.

When the teacher walks by, Violet excitedly asks, “Do you want to see my movie?” The teacher responds, “Sure, I would love to see your *program*,” attempting to reinforce the correct language. “What does the character in your program do?”

Violet responds enthusiastically, “I made a blinking pink cat!”

This case study was included to demonstrate the creative exploration some children may partake in when given ScratchJr with minimal instruction. When considering individual temperament, some interacted with the initial exposure to ScratchJr – a novel environment – with excitement and sustained attention towards their own self-created goals. While others were more cautious; they looked for guidance from peers and adults or found one aspect of the tool that they understood and stayed with for the duration of the first class.

Child 23/37. Classroom Pre8. How Accurate is On/Off-Task Even with Video Data?

Real-time audio and video capture of children while using ScratchJr was incredibly valuable. The videos accurately captured and preserved ScratchJr programs, the children’s attempts to tackle each lesson, and the emotional highs and lows during the ScratchJr programming process. The richness of these data is far beyond the analytic scope of this dissertation. The videos also captured behavior such as collaboration among peers and, occasionally, one child taking over another child’s computer. One future direction of this work will be to further

analyze the collected videos to assess the frequency of collaborations and computer take-overs, examining how peer interactions could influence learning to code. This type of behavior is not accurately accounted for, especially in On/Off-Task coding, and is outlined in the following case study.

Jessie and Belle were two girls in the Pre8 classroom. Jessie was 5.18 and Belle was 5.84. Jessie averaged a Time on Task percentage of 53.7%. Belle averaged a Time on Task percentage of 40.2%. For the Race Lesson, in particular, Belle's Time on Task is 47.1% and Jessie's Time on Task is 71.7%. Nearly every student scored higher than average for the Race Lesson, so, this is not too surprising. A 71.7% Time on Task Score is high (Race average was 59.7%), so the score alone is not particularly alarming. Instead, a note in Belle's file drew attention to this case. Belle managed to complete the Race Lesson, but, she did not do it herself. Jessie programmed most of it. So, during the minutes of blank screen time on Jessie's computer, where it looked as if she was "off-task" she was, in a way, still very much "on task" for her assigned lesson; she was programming characters to race on another student's computer.

This case highlighted a few issues. First, there needs to be a way to account for and give credit for collaboration, peer scaffolding, and peer programming in future studies. Second, the case again highlights the importance of the video recordings. Without the video and with scoring ScratchJr programs alone, it would have looked as if Belle had successfully created a Race program all on her own and researchers would not have had an accurate picture of her coding capabilities. Third, this case also demonstrated that the children in this

study reached out to one another for support. Although these interactions made the data analysis process messy, when children interacted with one another while interacting with the ScratchJr software, these social interactions perhaps enabled learning at a level beyond what the child could have achieved on his or her own.

Child 17. Classroom Pre16. Child Empowered through ScratchJr.

Prior to working with John, the researchers were told of his occasional inability to control his behavior. Just before one ScratchJr lesson in particular, he was in the principal's office after having thrown a chair at a teacher.

John was 5.60 years old. He scored a 0/60 on the HTKS pre-test assessment. John did fairly well with ScratchJr. He scored a 3/5 for Programming Score, a 1.40/2 for Goal Completion, and had a 50.4% average Time on Task Score. He particularly enjoyed the Race Lesson, scoring a 94.6% Time on Task Score.

Perhaps more importantly, John felt empowered. Classroom teachers quickly learned that John knew what he was doing in ScratchJr. Students got excited about John's programs. John was humble (sometimes) when he did not know how to do something and expressed enthusiasm when inquiring from his peers how they got an "awesome character" or "cool action" to happen. There is video evidence of teachers calling on John to come assist other students when they were struggling with ScratchJr. On several video recordings, you can hear in the background "Ask John, he knows how to do that."

The day he was in the principal's office the teacher negotiated to have him come back just for ScratchJr because it seemed to be the only time of the day he

really thrived. According to the teacher, John struggled with basic tasks, like writing his name, and the other children would make fun of him for it. With ScratchJr, he got to be the expert and many of the children looked to him for help. The videos showed many expressions of happiness he had during ScratchJr and the videos also evidence the teachers referring to John as a classroom expert with ScratchJr.

Child 7. Classroom Pre16.

Sally was one of the rare cases of a child who complained frequently about using ScratchJr. When Sally was not complaining (and occasionally when she was), she was coloring cats. Occasionally Sally would wander the classroom and inquire about other students' programs and once in a while interfere with them as well. Her combination of complaining and coloring cats eventually drew attention from researchers and teachers who were regularly redirecting her to program. Sally was 5.14 years old, her average Programming Score was 1.6, although, in the later lessons she scored higher (Sun, Dance, and Greet), in the 2s and 3s with adult help. Her average Goal Completion Score was a 0.5 and average Time on Task Score is 26.8%.

Sally's trajectory looked a bit like this: 1. In the Airplane Lesson, she colored cats; 2. In the Race Lesson, she colored cats; 3. In the Sun Lesson, she colored a combination of girls and cats with an underwater background. When a researcher joined Sally for the Sun Lesson and walked her through the program, she attempted to code. The researcher tried to teach her how to test ScratchJr programming blocks on her own. When the researcher left to assist other

students, Sally stopped coding; 4. During the Dance Lesson, Sally colored characters and dragged seemingly random programming blocks onto the scripting area; 5. For the Greet Lesson, Sally made progress by eventually connecting the send message and speech bubble blocks with adult assistance; and 6. For the Project Lesson, Sally worked towards making multiple pages with programming blocks on them, although Sally did not necessarily understand the programming blocks. This work did, however, show improvement over the prior lessons. Sally also programmed most of the Project Lesson independently.

Sally is included as an example of a child for whom the ScratchJr lessons alone did not seem to be enough to motivate her or engage her in the programming tasks. She was successful with programming tasks when she was provided significant support throughout the activities, guiding her through the correct programming blocks, and when she worked on the open-ended projects without adults constantly redirecting her away from the paint-editor. Sally's story highlights the need for future studies of curriculum design to investigate motivational factors that truly engage students in ScratchJr and other programming languages.

In sum, the case studies presented here profiled weak, consistent, and strong students coding with ScratchJr. Additional case studies were also presented to highlight specific areas of interest when interpreting the results of this study. For example, issues pertaining to scaffolding, motivation in the curriculum, unaccounted for learning difficulties and behavior issues, complications with video data collection, and the innovative ways children

approached learning to code with ScratchJr. These case studies provide a holistic and colorful picture of the kindergarten students to draw upon in the Discussion Chapter (Chapter Seven.) Chapter Seven also presents some limitations and future directions that need to be addressed in more detail.

Chapter Seven: Discussion

As I have noted, this study was a pilot investigation involving 38 students who maintained enrollment throughout the project. Although all results should be interpreted with caution, there are some exciting and promising possible relationships to explore in future work. This chapter will highlight some of the potential connections between self-regulation and coding skills in ScratchJr. This chapter will also discuss the many limitations to this study and where these results and limitations may be applied to future research.

This dissertation drew upon RDST to consider “to what extent does self-regulation have a role in learning to code with a novel computer programming software in kindergarten classrooms, and, does length of exposure to the programming software correlate with the post-test self-regulation scores of these kindergarten students?” A discussion and interpretation of the results follows.

Question 1 Results

The study reported here was part of a larger research project intended to create a developmentally appropriate programming software for young children ages 5-7 called ScratchJr. All areas of ScratchJr were designed with early childhood development in mind; however, this dissertation focused specifically on how self-regulation was accounted for in the design of ScratchJr. Page 51 contains a description of the components of ScratchJr specifically designed to account for young children’s developing self-regulation skills so that young children’s development self-regulation skills would not hinder their ability to use ScratchJr in the ways in which it appeared they were hindering use of Scratch.

Based on these parameters, it is possible that the design team of ScratchJr appropriately took into account the developmental needs of young children in terms of self-regulation. That is, it is possible that the team made the program equally accessible to all children regardless of whether they approached ScratchJr with a score of 0 or a score of 50 on the HTKS pre-test assessment. Children in this study were able to attempt to use and understand the ScratchJr interface and coding blocks with equivalent effectiveness, as measured by the Programming Score variable. This interpretation is positive in regard to the use of the software, in that it is designed for every student in the kindergarten classroom to use with equal success and that the software is developmentally appropriate across multiple dimensions, including self-regulation. Of course, the technology is not everything. The curricular supports and instruction also matter. Working to ensure the software is “developmentally appropriate” is working to ensure the software tool itself does not hinder use. Future assessments replicating these results will need to be carried out to assess the repeatability of these initial findings. Additional assessments of self-regulation and the components of self-regulation will also be added.

One possible way to interpret the results for Question 1, to what extent does self-regulation make a difference when learning to code, is that baseline levels of self-regulation, as measured by the HTKS assessment, did not make a difference when the children were attempting to code with ScratchJr. Yet, self-regulation did matter outside the scope of the ScratchJr interface when the children needed to be more goal-oriented. For example, since the development

team of ScratchJr designed the program to account for varying levels of self-regulation (e.g., limiting numbers, taking out “instant gratification” blocks) the children in the study had more equitable chances of success when using the software’s interface. However, in respect to the variables of the goal-directed component (Goal Completion Score) and the heavily relied upon attention and inhibitory control (Time on Task), the child’s baseline level of self-regulation may have had a greater impact on the outcome.

The possibility exists that the Goal Completion Score is tied too closely to the curriculum. The children, regardless of self-regulation, might have been able to understand the ScratchJr interface. Yet, self-regulation did seem to have a connection to Time on Task and Goal Completion Score. In regard to Goal Completion Score, perhaps it is not that the children with lower levels of self-regulation had more difficulty reaching goals, but instead, that the curricular goals themselves were not motivating enough for students. Substantive anecdotes where children were more successful at lessons about which they were more excited and enthusiastic (repeatedly revisiting Race programs or finding Greet program to be incredibly silly/funny) point to an argument for exploring the importance of engaging curriculum that is as motivating to students, as the overall Time on Task across lessons was 43.2%. Time on Task data may demonstrate that children can attend more to lessons they found interesting, like the Race lesson, but struggled to focus when the goal was too open-ended and the curriculum did not provide enough scaffolding (Project Lesson).

Question 2 Results

Although the results were not significant, the differences in self-regulation scores between pre-test and post-test for the Pre8 and Pre16 groups are of possible interest. There was an increase (albeit a non-significant one) in self-regulation score on the HTKS assessment for the group who learned to code with ScratchJr for sixteen lessons compared to the group who only learned to code for eight. If this relation could be confirmed in future research, perhaps there is a relationship where eight lessons were enough to gain some familiarity with coding concepts and understanding of the blocks in ScratchJr. However, it may be that a child needed somewhere between eight and sixteen lessons, once they mastered a general understanding of the interface, to begin to work on the problem-solving and debugging that would lead to more self-regulated behavior.

There needs to be some caution in interpreting these results because of unmeasured and uncontrolled for variables. Both groups were pre-test and post-tested at the same time to minimize other variables that could account for the difference between groups, but researchers were not present in the classrooms outside of the ScratchJr lessons.

Both teachers were in the same school and children from the same areas. However, it is possible that, each teacher could have had a unique impact on self-regulation. Future studies will include additional classrooms and control groups to help narrow the focus on the ScratchJr and self-regulation connection.

Curriculum and Lesson Design

The curriculum used in this study was likely a factor in how the children performed. For example, the Airplane Lesson was designed to be the least difficult lesson, as only three blocks were necessary and there were two possible ways to come to a solution (either change the number parameter or put multiple motion blocks). However, Race, Sun, and Dance Lessons all had higher average Programming Scores than the Airplane Lesson. Race was the most successful lesson, in terms of Programming Score average, Outcome Score, and Time on Task.

There could be several reasons for the score difference. First, the Airplane Lesson was the initial lesson, so, children had less exposure and practice with ScratchJr, thus potentially lower scores, even for an easier lesson. Second, the Race, Sun, and Dance lessons might have been more motivating to students. Perhaps, too, there was still a novelty effect when working with the Airplane Lesson and children were not yet focused on working towards the goal (lowest mean across lessons) or had not yet become familiar with the routines of the classroom and ScratchJr lessons.

The research team changed the curriculum as a result of this study. Instead of keeping the previous model of teaching lessons on a projected screen and having children use ScratchJr cards as a point of reference (see Appendix 1 for an example) the research team created tutorials called “Click It, Solve It, Make It” and integrated these tutorials into a new version of ScratchJr (see <http://www.scratchjr.org/teach> for examples). These tutorials use a scaffolded,

several-step process. First the child clicks on the targeted blocks, then the child solves some broken code, and finally the child makes the goal program on his or her own.

BRIEF

The BRIEF, meant as a triangulation assessment, did not triangulate with the HTKS data as expected (BRIEF scores correlating with HTKS scores, directly, as in a correlation between HTKS and BRIEF Global Score). Correlation results for the BRIEF indicate scores varied by classroom. These results are interesting, since no direct correlations were found between HTKS scores and BRIEF scores as expected for triangulation yet there does seem to be an interaction between the classroom a child was assigned to and his or her score on the BRIEF assessment, yet, no claims can be made in regard to a direct HTKS-BRIEF connection. A correlation was found between the classroom a child was assigned to and BRIEF score. Teachers took the BRIEF at the end of the ScratchJr study since the assessment required that teachers know the students for a minimum of three months prior to evaluating them. Since there were correlations between the BRIEF and the class, but not BRIEF and HTKS, two explanations seem possible.

Because the two classrooms had numerically but not statistically different HTKS post-test scores, it is possible that the BRIEF is also effected by self-regulation differences between the two groups since the BRIEF and post-test were given around the same time. A second explanation is that there was something about the ways in which the teachers approached filling out the BRIEF that led to

the differences between classrooms. However, neither the “negativity” nor “inconsistency” indices within the BRIEF scoring system were flagged for either teacher.

There were other potential issues with the BRIEF. Any of the following might be true: the BRIEF is too clinical for a general school research setting, teachers found they did not have enough data about their students to honestly and accurately answer the questions after just three months of knowing the children, the questions were not completely applicable to kindergarten (feedback the researcher did receive), or teachers preconceived notions of the child may not have been accurate. In the future, additional standardized measures of self-regulation and EF will be used to triangulate one with another, administered from the parent, child, and teacher perspectives.

Limitations, Future Directions, and Conclusions

There are numerous inter-related limitations in this study primarily regarding the technology, participants, and missing data that resulted. This study is exploratory. As the limitations lead directly to potentially interesting areas of future study, both limitations and future directions are reported together in this section.

Limitations

The children in this study were instructed on how to correctly handle their individual laptops during their first lesson of ScratchJr. Unfortunately, students did not always follow the rules (e.g., keeping laptops on the desk, not pulling out the mice, not banging on the keys, etc.). As a result, sometimes video data were

lost when a child hit a key or series of keys that exited him or her from the Camtasia recording program. Researchers in the classroom tried to watch for technical glitches and restart the recordings but were not always successful.

Another issue with recording arose because the computers were transported directly between two classrooms at different schools. The first classroom had limited power outlets and the second classroom had no power outlet access due to the table set-up in the room. Occasionally, computers would lose battery life mid-lesson and the video would also be lost or only a partial recording would be saved. Another recording issue arose when too much memory was in use either due to too many saved files (the computers were used for multiple lab projects) or to the child closing and opening multiple programs. If too much memory was used, the video files sometimes became corrupted and unusable, or the ScratchJr software froze to a point of being unusable and Camtasia had to be closed. Researchers tried to only use the newest laptops to avoid the memory issue, but, due to limited resources, all laptops were used at different points in time. Finally, a video or two were just too blurry to use, the result, perhaps, of sticky fingers on the camera lens.

Missing videos were the primary cause of missing variable data throughout the reporting of the results. A secondary cause of missing data was movement of students in and out of the classroom due to absences, moves to new schools, refusal to participate in pre-testing activities, and parents not consenting to data collection via video. There was also one day when the servers that hosted ScratchJr went down, locking the ScratchJr program in the paint editor. Data

from this lesson were not used and children repeated that day's lesson. Since children were primarily exposed to blank screens and/or the paint editor, it is not believed that day had any impact on the children learning to code. Still, there were other days when the servers would be slow or crash and Camtasia would need to be stopped (resulting in shortened or no videos) or children would get "stuck" in/on a particular area of the ScratchJr interface. These technical glitches were accounted for in on/off-task coding (children were not coded as "off" task if they were not working due to a technical glitch.) Still, being stuck and unable to code may have impacted the number of coding blocks a child was able to attempt/master or how far the child progressed through the curriculum.

In future studies, the researcher will better account for technical glitches in video data collection due to rampant nature of the technical problems in this study. Updated and newer screen capture software tools will benefit future work as Camtasia has obvious limitations (e.g., cost, cannot be used on iPads, memory use). In addition, the newest version of ScratchJr is iPad based, rather than web-based, which will result in fewer problems based on server error or Internet connections in the classroom.

Demographics

There are also missing demographic data. However, the missing data do not come from video or data collection errors but rather from a calculated decision of the researchers. In order to get as much compliance as possible from the families involved (as it stands, a third control group did not participate in the fall session of the study), the decision was made to ask for minimal information

on the consent documents. We asked for consent for participation, consent for audio/video recording, consent for use of audio/video recordings, and the child's date of birth. These consent forms were then translated into Spanish and Portuguese. It would have been beneficial to have more specific data for each child on socioeconomic demographics, as there is a known divide in technology use based on home income as well as a deficit in self-regulation scores for both low-SES children and children with ELL status. Since a demographic background of the school was known, a broad understanding of why the sample may have a floor effect around the HTKS variable was deduced.

It also would have been beneficial to explore the data in this study based on race/ethnicity as there are large gaps in STEM fields along racial/ethnic divides. Furthermore, it would have been beneficial to track children's performance on ScratchJr. in relation to their exposure to computers, iPads, Kindles, or even learning to code (recall the anecdote in the Case Study Chapter where one parent described teaching his child original Scratch). Post-hoc requests to teachers for more specific classroom demographics were not fulfilled.

Future studies should examine learning to code with ScratchJr, and potentially other programming languages, with additional focus on specific demographic information, particularly in relation to socioeconomic status, race/ethnicity, and home/school digital technology exposure. There potential for longitudinal research in tracking children who learn to code in kindergarten to see if they are more likely to take higher level math, science, or computing courses, major in STEM disciplines in college, or go on to a STEM career when compared

to peers who did not learn to code at a young age. This research would be particularly helpful for under-represented groups in these areas. In the shorter term, longitudinal tracking of students may be used to look at students who learn to code in Kindergarten and those who do not to compare performances on math and literacy tasks and assessments in later grades of elementary school.

Ways of Coding Variables

The HTKS assessment is a standardized assessment but with the floor effects apparent in this sample that tend to occur for low-SES and high-ELL populations. For the purpose of this study, the researcher created the Programming Score, Goal Completion Score, and Time on Task Score variables. When working with innovative new technologies it is difficult to find standardized ways of quantifying outcomes. It is possible that knowledge of block understanding was under-scored in the Programming Score assessment. However, underestimating the block understanding (by giving a child an attempt, or 1, rather than an understood, or 2, if in doubt) was a better approach than potentially making overly generous or overly reaching claims about what a child could achieve with a technological tool that is new to him or her. In turn, the Goal Completion Score variable is structured based on the goal of the lesson. The outcomes in goal-directed behavior might look different if the researchers were to collect systematic data on the child's intended goal and the faithfulness of his or her program to that intended goal. Even so, with the semi-open-ended Project Lessons, it proved difficult to elicit concrete goals from kindergarten students or to analyze whether children carried out the goals they stated. Finally, for Time on

Task, the coding was not as precise as it could have been. Coders watched the videos and recorded by hand the actions and timestamps of the children's transitions between "on" and "off" task behavior. Future studies will explore more precise ways of examining Time on Task and evaluating data of this type. Regardless, since the same method of analysis was applied to all children, it served its purpose for this pilot study in making comparisons between lessons and in measuring differences between children. Additional standardized measures will be sought in the future in order to more concretely study the concepts of programming knowledge, goal-oriented behavior and time spent on task.

Scaffolding

Scaffolding during the lessons of the ScratchJr study reported in this dissertation varied by type, day, child, classroom, and other factors. For example, a Monday classroom might have had the primary researcher, two research assistants, the lead classroom teacher, and the assistant classroom teacher compared, whereas a Thursday classroom might have had only the primary researcher and one research assistant. All research assistants and teachers received introductory lessons to ScratchJr. They received instructions to offer students scaffolded prompts only, and not offer specific directions. A scaffolded prompt might be, "Which color blocks would you need if you wanted to make your character move?" whereas a specific direction might be, "Drag that jump block down here." However, in the midst of a sometimes chaotic classroom setting, these rules were not always followed. In addition, some children were much more vocal about seeking assistance from adults than others. Scaffolding

likely had some impact on how well children performed, especially for children like Jake and Sally for whom an adult's presence was vital to their staying on task to achieve goals. Nonetheless, it is beyond the scope of this study to analyze a scaffolding variable. Future work will focus on imbedded scaffolding in the curriculum and reduce the need for multiple adults to be present in the classroom or home to use ScratchJr. Future work may also include comparison of similar curriculum lessons from both the fall ScratchJr curriculum (ScratchJr cards) and the spring ScratchJr curriculum (Click It, Solve It, Make It).

Future Directions

The future directions of work based on the results and limitations in this dissertation should involve a focus on exploring young children's self-regulation when they are learning to code, young children's use of ScratchJr, and general parameters of young children learning to code.

First, this study could be expanded by increasing the number and types of classrooms and students who participate in similar studies through collecting more complete demographic data on students. In addition, tracking students for longer periods of time, would create a more robust picture of who learns to code, how they learn to code, and potential learning outcomes that could translate to other areas and persist over time. Second, self-regulation could continue to be a focus of the research around the ScratchJr programming language. It could be examined further by including additional measurements at both the student and teacher/parent level for the purpose of triangulating results across multiple measures or delving more deeply into specific areas such as attention or working

memory. Third, exploration between self-regulation and software tools beyond ScratchJr. may prove to be a promising future step. There are multiple new programming languages on the market for young children. Are they developmentally appropriate? By recording children's use of these tools in similar ways to this study, researchers may be able to determine if children are effectively learning new programming languages. Researchers may also be able to evaluate whether or not the languages are developmentally appropriate. Furthermore, these studies might contribute to literature discussing ways to evaluate young children learning to code more succinctly.

In addition, by working with ScratchJr or additional programming languages for children, connections may be explored between learning to code and children with learning differences. Anecdotally, educators have expressed that certain children with difficulties in other areas of the classroom and school day are more engaged with coding and technology-based activities than other aspects of the daily curriculum. In a case study reported here, a teacher requested that a student with behavior difficulties (no diagnosis given) be returned from the principal's office to the classroom for ScratchJr time because this was the only time of the day he seemed to thrive. The examination of technology use and learning to code for children with learning difficulties and learning differences is a vast area of research and is far beyond the scope of this current project. This area of exploration, however, is one that should be explored as it is mentioned frequently when the work in this dissertation, and similar work, is presented.

Although no differences were found between girls and boys in this study, there is a gender divide pertaining to which students pursue STEM-based careers in the future. A gender divide is also evident in studies (Matthews, Pontiz, & Morrison, 2009) regarding differences in self-regulation between boys and girl. As such, future studies should pay particular attention to differences between boys and girls in outcome and engagement when learning to code with ScratchJr.

Finally, this work provided some indication that learning to code may improve self-regulation skills in kindergarten. Future work should be conducted to continue to explore self-regulation as well as to observe connections to other areas of the early childhood curriculum including early math and early literacy skills. Learning to code in the context of robotics has already been connected to sequencing, an early literacy skill (Kazakoff & Bers, 2012; Kazakoff, Sullivan, & Bers, 2013). Additional analysis of ScratchJr data will attempt to determine whether there are similar connections to sequencing as a first step in connecting ScratchJr coding to early literacy. There are still many other avenues to exploring the connection of traditional areas of the early childhood curriculum to less traditional areas such as coding, computational thinking, problem solving, and digital literacy skills.

Implications

The results from this study yielded many avenues of future work in exploring the intersection of learning to code and self regulation, curriculum design around learning to code, and analysis of individual trajectories of collaboration and problem-solving when learning to code. Further in-depth

examination of the video data collected may be used as a first step toward analyzing collaboration between students and problem-solving trajectories.

Future work in the area of learning to code in early childhood and evaluating educational technologies for early childhood may focus on what is and is not developmentally appropriate in terms of interface design for the target age group (in this case, five to seven year olds). Some of the evaluation criteria outlined in this dissertation may be used for this evaluation purpose, such as the changes made between Scratch and ScratchJr. Despite being a small study, the changes made between Scratch and ScratchJr for self-regulation purposes appear to have made a difference in learning to code for the students in this study because the types of projects imagined in initial pilot studies with Scratch (e.g., making characters race) are now a reality with ScratchJr.

Ideally, a future study would include multiple classrooms in multiple schools in diverse locals (in terms of both SES and racial/ethnic demographics of the students). Background data would be collected from parent(s)/guardian(s) pertaining to SES, race/ethnicity, technology use at home, known learning difficulties, child's interest in technology, and any technology enrichment programs in which the child was/is enrolled. Data would also be collected from teachers regarding time teaching, technology integration in school/classroom, and comfort with technology use in classroom. Self-regulation assessments of the children would be given to both parents and teachers to complete. One-on-one self-regulation assessment games would be run directly between the researchers and the children pre-computer programming intervention, at a mid-point, and

post-computer programming intervention. Classrooms would be randomly assigned to varying lengths of computer programming interventions to be able to assess the impact of length of exposure to learning to code on learning and developmental outcomes. Audio and video recordings of the classroom dynamics would occur and be coded for differences in teacher approaches to instruction, collaboration between students, and classroom dynamics. Consent to track students longitudinally would be collected and follow-up surveys would be sent to students and their teachers each quarter, semester, or year to assess progress in areas such as math skills, literacy skills, self-regulation, STEM interest, and STEM pursuits.

In Conclusion

This study demonstrated that a group of 38 kindergarten students, to varying degrees (likely based on a variety of variables that could not be assessed in the scope of this study), actively engaged with a new, developmentally appropriate programming language, ScratchJr. Children in this study were all able to select and program characters. There appeared to be no differences in the ability to attempt and understand the use of the ScratchJr programming blocks and interface components based on children's baseline self-regulation levels.

Furthermore, it seemed there was potentially a relationship between learning to code and increased self-regulation skills, but this relationship was not significant given the small sample. Furthermore, mediating variables such as classroom culture, home demographics, and individual child temperament cannot be ruled out as factors in the relationship.

When software is designed well and designed to be developmentally appropriate, children will actively engage with the digital tool. Without question, individual variability in use will remain but, overall, the tool should have a low floor for easy and engaging access. The work presented here demonstrated, through the analysis of the Programming Score variable, that ScratchJr may be developmentally appropriate in the targeted area of self-regulation. ScratchJr was also designed to be developmentally appropriate in areas, such as math and literacy, but, those areas were not specifically evaluated.

Children with varied levels of self-regulation were able to understand the ScratchJr interface at roughly the same levels. Even children who clustered at the floor effect for self-regulation had a wide range of scores across the three programming variables, further demonstrating that low self-regulation was not necessarily a barrier to entry for use and understanding of coding with ScratchJr. However, although it was a small difference, there did seem to be some differences overall in goal-oriented behavior and time on task scores for the open-ended, less scaffolded Project Lesson based on initial levels of self-regulation meaning that the children who had higher levels of self-regulation were more goal-oriented, which is consistent with expectations and prior work. Furthermore, the children, in general, performed better across all variables on the more entertaining rather than the 1) more open-ended/personally-meaningful or 2) easiest lessons. This pattern may indicate a need for a more engaging, scaffolded, structured curriculum around digital tools when introducing the digital tools in early childhood classrooms. Overall, this work was a promising step in designing

and evaluating the development of programming languages to be developmentally appropriate and universally designed for use by children in kindergarten.


List of Appendices

Appendix 1. ScratchJr Card for Airplane Lesson.	123
Appendix 2. ScratchJr Interface Checklist.	124
Appendix 3. ScratchJr Design Considerations and Rationale beyond Self- Regulation.	125



Appendix 1. ScratchJr Card for Airplane Lesson.

SCRATCHJR Can I Make My Plane Fly Across the U.S.A?


1. Choose Background



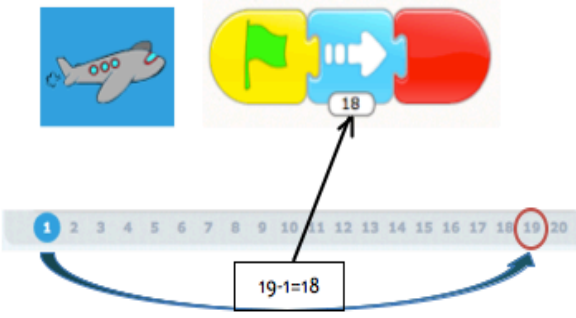
2. Choose Character



3. Move Character to Start Place












































4. Make Program



Appendix 2. ScratchJr Interface Checklist.

ScratchJr Fall 2012 Video Coding Sheets

Video Coder:	Video #
CODING SCHEME	Time Comments (continue on back if needed)
Didn't Use the Block - Leave Blank	-----
Attempted to Use the Block (put a dot in the box as soon as the child uses a block)	 -----
Understands the Block (put a check mark over the dot when the child understands block. Write the timestamp.)	 15:28 -----

BLOCKS & INTERFACE	Code	BLOCKS & INTERFACE	Code	BLOCKS & INTERFACE	Code	BLOCKS & INTERFACE	Code
							
Start on Green Flag		Wait		Speech Bubble		Choose A Character	
							
Start on Clicked		Stop		Change Words in Speech Bubble		Draw Own Character	
							
Start on Touched		Repeat		Grow		Color Fill a Character	
		Change a # in a Control Parameter					
Start on Message Rec'd		Change a # in a Control Parameter		Shrink		Choose A Background	
							
		Set Speed		Set Size Equal		Draw Own Background	
				Change a Looks # Parameter			
		Send Message		Change a Looks # Parameter		Add Text	
		Change Envelope Colors or Speed using					
		Change Envelope Colors or Speed using		Hide		Color Text	
							
		Sounds		Show		Resize Text	
							
		Repeat Forever				Grid On and/or Off	
							
		Go to Next Page				Add a Page	
							
		End				Delete A Character	
		Drag & Drop Characters to New Pages				Multiple Scripts on Character	
Go Home		Drag & Drop Characters to New Pages				Multiple Scripts on Character	
Change a # in a motion Parameter		Drag & Drop Scripts to New Characters				Characters with Scripts	

Appendix 3. ScratchJr Design Considerations and Rationale beyond Self-Regulation.

ScratchJr	Rationale
Blocks primarily icon-based with words displayed on "hover/tap"	The icon-based interface is more intuitive for use by children who do not yet read than the original Scratch word-based interface. However, the interface does have the option to display the words of the programming blocks for those children who do read and for adults who may be assisting students. Furthermore, there is a adding text feature where a child may create his or her own letters, words, and sentences.
Large programming blocks	The icons of the programming blocks in ScratchJr are larger than in the original Scratch, which is a better fit with the motor skills of younger children (i.e., more surface area to target with mouse or finger).
Easy Fill Characters	The colors on characters can be easily changed to make characters more personally relevant (e.g., hair, eyes, skin, clothes). The free drawing of characters and backgrounds is also an option.
Story Pages	The ability to add up to four pages to a program, called "Story Pages," exists for ease of making cohesive books and building connections to early literacy
Graph (w/ on/off switch)	A number-based graph overlays the stage for assisting with number sense and understanding where the characters will travel or have travelled. The graph can be turned off if it is distracting when playing a program.

Endnotes

¹Classroom Pre16 was exposed to this lesson twice since the server crashed during the first exposure and the children were not able to program their own projects. For the majority of students, the second day of “Airplane” will be assessed, unless a video only exists for Day 1. Double exposure will likely not impact the outcome, as the computer glitch left most children stuck in the paint editor, not the programming screen.

²Teachers gave out the consent forms in English plus a secondary language to the families to whom they believe would need the translated form. Teachers were provided with a starter set of forms and photocopied them as needed.

References

- Baumeister, R. F., & Vohs, K. D. (2004). *Handbook of self-regulation: Research, theory, and applications*. New York, NY: Guilford Press.
- Blair, C. (2002). School readiness: Integrating cognition and emotion in a neurobiological conceptualization of child functioning at school entry. *American Psychologist, 57*, 111-127.
- Brennan, K., & Resnick, M. (2012, April). *New Frameworks for Studying and Assessing the Development of Computational Thinking*. Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.
- Brown, A. L. & French, L. A. (1976). Construction and regeneration of logical sequences using causes or consequences as the point of departure. *Child Development, 47*(4), 930 – 940.
- Bell, P. (2004). On the theoretical breadth of design-based research in education. *Educational Psychologist, 39*(4), 243-253.
- Bers, M. (2007). Positive Technological Development: Working with computers, children, and the Internet. *MassPsych, 51*(1), 5-7, 18-19.
- Bers, M. U. (2008). *Blocks to Robots: Learning with technology in the early childhood classroom*. New York: Teacher's College Press.
- Bers, M. (2010) Beyond computer literacy: Supporting youth's positive development through technology. *New Directions for Youth Development, 128*, 13 – 23.

- Bers, M. (2012). *Designing Digital Experiences for Positive Youth Development: From playpen to playground*. New York, NY: Oxford University Press.
- Bers, M. & Horn, M. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. In I. R. Berson & M. J. Berson (Eds.), *High-tech tots: Childhood in a digital world*. pp. 49-70. Greenwich, CT: Information Age Publishing.
- Blair, C. & Razza, R. P. (2007). Relating effortful control, executive function, and false belief understanding to emerging math and literacy ability in kindergarten. *Child Development, 78*, 647-663.
- Bull, R. & Scerif, G. (2001). Executive functioning as a predictor of children's mathematics ability: Inhibition, switching, and working memory. *Developmental Psychology, 19*(3), 273-293.
- Bus, A.G., & Newman, S.B. (2009). *Multimedia & literacy development: Improving achievement for young learners*. New York: Routledge.
- Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education, 22*(4), 711 – 722.
- Chiong, C., & Shuler, C. (2010). *Learning: Is there an app for that?* New York, NY: The Joan Ganz Cooney Center at Sesame Workshop.
- Clements, D.H. (1986). Effects of LOGO and CAI environments on cognition and creativity. *Journal of Educational Psychology, 78*, 309-318.
- Clements, D. H. (1987). Longitudinal study of the effects of LOGO programming on cognitive abilities and achievement. *Journal of Educational Computing*

Research, 3, 73–94.

Clements, D. H. (1999). The future of educational computing research: the case of computer programming. *Information Technology in Childhood Education Annual, 147-179.*

Clements, D. H., & Nastasi, B. K. (1992). Computers and early childhood education. In Gettinger, M., Elliott, S. N., & Kratochwill, T. R. (Eds.), *Advances in school psychology: Preschool and early childhood treatment directions* (pp. 187–246). Hillsdale, NJ: Lawrence Erlbaum Associates.

Clements, D.H., Battista, M.T. & Sarama, J. (2001) LOGO and Geometry, *Journal for Research in Mathematics Education Monograph Series, 10.*

Clements, D. H., & Sarama, J. (2002). The role of technology in early childhood learning. *Teaching Children Mathematics, 8, 340-343.*

Clements, D.H., Sarama, J., Unlu, F., Layzer, C. (2012, March). The efficacy of an intervention synthesizing scaffolding designed to promote self-regulation with an early mathematics curriculum: Effects on executive function. Presentation at *Society for Research on Educational Effectiveness (SREE)*. 8 – 12 March 2012. Washington, DC.

Clements, D. H., & Swaminathan, S. (1995). Technology and school change: New lamps for old? *Childhood Education, 71, 275-281.*

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher, 32(1), 9-13.*

Common Sense Media (2011). *Zero to eight: Children's media use in America.*

Retrieved from <http://www.commonsensemedia.org/research/zero-eight->

childrens-media-use-america

- Copple, C. & Bredekamp, S. (2009). *Developmentally appropriate practice in early childhood programs serving children from birth through age 8*. Washington, DC: National Association for the Education of Young Children.
- Cordes, C., & Miller, E. (2000). *Fool's gold: A critical look at computers in childhood*. College Park, MD: Alliance for Childhood.
- Crook, C. (1998). Children as computer users: the case for collaborative learning. *Computers & Education, 30*(3-4), 237-247.
- Cunha, F. & Heckman, J. (2007). The technology of skill formation. *NBER Working Paper No. 12840*. Retrieved from <http://www.nber.org/papers/w12840>
- Davidson, C. N. (2011). *Now You See It: How the brain science of attention will transform the way we live, work, and learn*. New York, NY: Viking.
- Degelman, D., Free, J.U., Scariato, M., Blackburn, J.M., & Goldent, T. (1986). Concept learning in preschool children: Effects of a short-term LOGO experience. *Journal of Educational Computing Research, 2*, 199-205.
- Diamond, A. (2002). Normal development of prefrontal cortex from birth to young adulthood: Cognitive functions, anatomy, and biochemistry. In D. T. Stuss & R. T. Knight (Eds.), *Principles of frontal lobe function* (pp. 466–503). London, England: Oxford University Press.
- Diamond, A. (2010). The evidence base for improving school outcomes by addressing the whole child and by addressing skills and attitudes, not just

content. *Early Education and Development*, 21, 780-793.

- Diamond, A & Lee, K. (2011). Interventions shown to aid executive function development in children 4 to 12 years old, *Science*, 333(6045), 959-964.
- Druin, A. (1998). *The Design of Children's Technology* (Ch. 2-3). San Francisco: Morgan Kaufmann.
- Elkind, D. (1986). Formal education and early childhood education: An essential difference. *Phi Delta Kappan*, 71, 631-642.
- Espy, K.A., McDiarmid, M.M., Cwik, M.F., Stalets, M.M., Hamby, A. & Senn, T.E. (2004). The contribution of executive functions to emergent mathematic skills in preschool children. *Developmental Neuropsychology*, 26(1), 465-486.
- Flannery, L.P., Kazakoff, E.R., Bontá, P., Silverman, B., Bers, M.U., and Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*. ACM, New York, NY, USA, 1-10.
- Fleer, M. (1999). The science of technology: Young children working technologically. *International Journal of Technology and Design Education*, 9, 269-291.
- Fletcher-Flinn, C. M., & Gravatt, B. (1995). The efficacy of computer assisted instruction (CAI): A meta-analysis. *Journal of Educational Computing Research*, 12(3), 219-242.
- Ford, D.H., & Lerner, R.M. (1992). *Developmental systems theory: An integrative approach*. Newbury Park, CA: Sage.

- Forman, G. (1988). Making intuitive knowledge explicit through future technology. In G. Forman & P. B. Pufall, Eds. *Constructivism in the Computer Age*. New Jersey: Lawrence Earlbaum.
- Forman, G. & Pufall, P. B. (1988). *Constructivism in the Computer Age*. New Jersey: Lawrence Earlbaum.
- Freelon, D. (2013). ReCal OIR: Ordinal, interval, and ratio intercoder reliability as a web service. *International Journal of Internet Science*, 8(1), 10-16.
- Garon, N. Bryson, S.E. & Smith, I.M. (2008). Executive Function in Preschoolers: A Review Using an Integrative Framework. *Psychological Bulletin*, 134(1), 31-60.
- Gee, J.P. (2008). *Getting Over the Slump: Innovation Strategies to Promote Children's Learning*. New York, NY: The Joan Ganz Cooney Center at Sesame Workshop.
- Gee, J.P. (2010). *New Digital Media and Learning as an Emerging Area and 'Worked Examples' as One Way Forward*. Cambridge, MA: MIT Press.
- Gee, J.P. (2013a). *Anti-education era: Creating smarter students through digital learning*. New York, NY: Palgrave Macmillan.
- Gee, J.P. (2013b) *Digital Media and Learning: A Prospective Retrospective*
Retrieved from
http://www.dies.uniud.it/tl_files/utenti/crisci/Humans%20learn%20from%20experience.pdf
- Gioia, Isquith, Guy, Kenworthy, (2000). *Behavior Rating Inventory of Executive Function*. Odessa, FL: Psychological Assessment Resources.

- Gropen, J., Clark-Chiarelli, N., Hoisington, C., Ehrlich, S.B. (2011). The importance of executive function in early science education. *Child Development Perspectives, 5*(4), 298-304.
- Gutnick, A. L., Robb, M., Takeuchi, L., & Kotler, J. (2010). *Always connected: The new digital media habits of young children*. New York: The Joan Ganz Cooney Center at Sesame Workshop.
- Hayes, A.F. & Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures, 1*(1), 77-89.
- Haugland, S. W. (1992). The effect of computer software on preschool children's developmental gains. *Journal of Computing in Childhood Education, 3*(1), 15-30.
- Heckman, J.J. (2006). Skill formation and the economics of investing in disadvantaged children. *Science, 312*(5782), 1900-1902.
- Heckman, J.J. & Masterov, D.V. (2004). The productivity argument for investing in young children. *Applied Economic Perspectives and Policy, 29*(3), 446-493.
- Hmelo-Silver, C.E., Duncan, R.G., & Chinn, C.A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and Clark (2006). *Educational Psychologist, 42*(2), 99-107.

- Huebner, G. (2013). *How you can change programming education*.
<http://www.surfscore.com/2013/10/30/how-you-can-change-programming-education.html>
- Jelicic, H., Theokas, C., Phelps, E., & Lerner, R. M. (2007). Conceptualizing and measuring the context within person \longleftrightarrow context models of human development: Implications for theory, research and application. In T. D. Little, J. A. Bovaird, & N. A. Card (Eds.). *Modeling contextual effects in longitudinal studies* (pp. 437-456). Mahwah, NJ: Erlbaum.
- Jenkins, H. (2006). *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. Chicago, IL: The John D. and Catherine T. MacArthur Foundation.
- Jenkins, H. (2009a). *Critical Information Studies for a Participatory Culture (Part Two)*. Retrieved from
http://henryjenkins.org/2009/04/what_went_wrong_with_web_20_cr_1.html
- Jenkins, H. (2009b). *Confronting the Challenges of a Participatory Culture: Media education in the 21st century*. Cambridge, MA: MIT Press.
- Kamps D., Abbott M., Greenwood C., Wills H., Veerkamp M., Kaufman J. (2008). Effects of small-group reading instruction and curriculum differences for students most at risk in kindergarten: Two-year results for secondary- and tertiary-level interventions. *Journal of Learning Disabilities, 41*(2), 101-114.
- Kazakoff, E.R., & Bers, M. (2012). Programming in a robotics context in the

kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia*, 21(4), 371-391

Kazakoff, E.R., Sullivan, A. & Bers, M. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, 41(4), 245-255.

Klahr, D. & Carver, S. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, 20, 362-404.

Lee, I., Martin, F., Denner, J., Coulter, B., & Allan, W., et al. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32-37.

Lerner, R.M. (2002). *Concepts and theories of human development* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Lerner, R.M. (2006). Developmental science, developmental systems, and contemporary theories. In W. Damon & R.M. Lerner (Editors-in-Chief) & R.M. Lerner (Vol. Ed.), *Handbook of child psychology*. Vol. 1. Theoretical models of human development (6th ed., pp. 1-17). Hoboken, NJ: Wiley.

Lerner, R. M., Lerner, J. V., Almerigi, J., Theokas, C., Phelps, E., Gestsdottir, S. Naudeau, S., Jelicic, H., Alberts, A. E., Ma, L., Smith, L. M., Bobek, D. L., Richman-Raphael, D., Simpson, I., Christiansen, E. D., & von Eye, A. (2005). Positive youth development, participation in community youth development programs, and community contributions of fifth grade adolescents: Findings from the first wave of the 4-H Study of Positive Youth Development. *Journal of Early Adolescence*, 25(1), 17-71.

- Lerner, R. M., Lerner, J. V., Bowers, E. P., & Geldhof, G. J. (In press). Positive youth development and relational developmental systems. In W. F. Overton & P. C. Molenaar (Eds.), *Theory and Method*. Volume 1 of the *Handbook of Child Psychology and Developmental Science* (7th ed.). Editor-in-chief: R. M. Lerner. Hoboken, NJ: Wiley.
- Liao, Y.-K., & Bright, G. (1991). Effects of computer-assisted instruction and computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research*, 7(3), 251–268.
- Madill, H., Campbell, R.G., Cullen, D. M., Armour, M.A., Einsiedel, A.A., Ciccocioppo, A.L.,...Coffin, W.L. (2007). Developing career commitment in STEM-related fields: myth versus reality. In R.J. Burke, M.C. Mattis, & E. Elgar (Eds.), *Women and Minorities in Science, Technology, Engineering and Mathematics: Upping the Numbers* (pp. 210 – 244). Northhampton, MA: Edward Elgar Publishing.
- Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4), 16.
- Markert, L. R. (1996). Gender related to success in science and technology. *The Journal of Technology Studies*, 22(2), 21-29.
- Matthews, J.S., Pontiz, C.C., & Morrison, F. J. (2009). Early gender differences in self-regulation and academic achievement. *Journal of Educational Psychology*. 101(3), 689-704.

- McClelland, M.M., Acock, A.C., Piccinin, A., Rhea, S.A., & Stallings, M.C. (2013). Relations between preschool attention span-persistence and age 25 educational outcomes. *Early Childhood Research Quarterly, 28*(2), 314-324.
- McClelland, M. M., & Cameron, C. E. (2011). Self-regulation and academic achievement in elementary school children. In R. M. Lerner, J. V. Lerner, E. P. Bowers, S. Lewin-Bizan, S. Gestsdottir, & J. B. Urban (Eds.), *Thriving in childhood and adolescence: The role of self-regulation processes. New Directions for Child and Adolescent Development, 133*, 29–44.
- McClelland, M.M., & Cameron, C.E. (2012). Self-regulation in early childhood: Improving conceptual clarity and developing ecologically valid measures. *Child Development Perspectives, 6*(2), 136-142.
- McClelland, M. M., Cameron, C. E., Connor, C. M., Farris, C. L., Jewkes, A. M., & Morrison, F. J. (2007). Links between behavioral regulation and preschoolers' literacy, vocabulary and math skills. *Developmental Psychology, 43*, 947–959.
- McClelland, M. M., Geldhof, G. J., Cameron, C. E., & Wanless, S. B. (In press). Development and self-regulation. In W. F. Overton & P. C. Molenaar (Eds.), *Theory and Method. Volume 1 of the Handbook of Child Psychology and Developmental Science* (7th ed.). Editor-in-chief: R. M. Lerner. Hoboken, NJ: Wiley.

- McClelland, M. M., Morrison, F. J., & Holmes, D. L. (2000). Children at-risk for early academic problems: The role of learning-related social skills. *Early Childhood Research Quarterly, 15*, 307–329.
- McClelland, M. M., Ponitz, C. C., Messersmith, E., & Tominey, S. (2010). Self-regulation: The integration of cognition and emotion. In R. M. Lerner (Editor-in-Chief) & W. F. Overton (Vol. Ed.), *The Handbook of Life-Span Development. Vol. 1: Cognition, Neuroscience, Methods* (pp. 509–553). Hoboken, NJ: Wiley.
- McClelland, M.M., & Wanless, S.B (2012) Growing up with assets and risks: The importance of self-regulation for academic achievement. *Research in Human Development, 9*(4), 278-297.
- Medvin, M., Reed, D., Behr, D., & Spargo, E. (2003). Using technology to encourage social problem solving in preschoolers. In G. Marshall & Y. Katz (Eds.), *Learning in school, home, and community: ICT for early and elementary education* (pp. 13-20). Boston: Kluwer Academic Publishers.
- Metz, S.S. (2007). Attracting the Engineering of 2020 Today. In R. Burke and M. Mattis (Eds.), *Women and Minorities in Science, Technology, Engineering and Mathematics: Upping the Numbers* (pp. 184-209). Northampton, MA: Edward Elgar Publishing.
- Miller, G. E., & Emilhovich, C. (1986). The effects of mediated programming instruction on preschool children's self-monitoring. *Journal of Educational Computing Research, 2*, 283 – 297.

- Mioduser, D. & Levy, S. (2010). Making sense by building sense: Kindergarten children's construction and understanding of adaptive robot behaviors. *International Journal of Computers for Mathematical Learning*, 15(2), 99-127.
- Mioduser, D., Levy, S. & Talis, V. (2009). Episodes to scripts to rules: concrete- abstractions in kindergarten children's explanations of a robot's behaviors. *International Journal of Technology and Design Education*, 19(1), 15-36.
- Moll, L.C. (2014). *L.S. Vygotsky and education*. New York, NY: Routledge.
- Muller, A. A., & Perlmutter, M. (1985). Preschool children's problem-solving interactions at computers and jigsaw puzzles. *Journal of Applied Developmental Psychology*, 6, 173 - 186.
- NCTE (2008). *The definition of 21st century literacies*. Retrieved from <http://www.ncte.org/governance/literacies>
- Overton, W. F. (2010). Life-span development: Concepts and issues. In R.M.Lerner (Ed). *Handbook of Life-span Development* (pp. 1-29). Hoboken, NJ: Wiley.
- Overton, W. F. & Mueller, U. (2012). Chapter 2. Metatheories, theories, and concepts in the study of development. In R.M. Lerner, M.A. Easterbrooks, & J. Mistry (Eds.), *Handbook of Psychology: Vol. 6. Developmental Psychology* (2nd ed.) (pp. 19 – 58). Hoboken, NJ: Wiley.
- Palfrey, J. & Gasser, U. (2008). *Born Digital: Understanding the first generation of digital natives*. New York, NY: Basic Books.
- P21 (2009). *P21 Framework Definitions*. Retrieved from

http://www.p21.org/storage/documents/P21_Framework_Definitions.pdf

- Paris, A. H. & Paris, S. G. (2003). Assessing narrative comprehension in young children. *Reading Research Quarterly*, 38(1), 36 – 76.
- Papert, S. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 161–191). Norwood, NJ: Ablex.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books: New York, NY.
- Piaget, J. (1970). *Genetic epistemology*. New York: W. W. Norton and Company.
- Piaget, J. (2000). Commentary on Vygotsky's criticisms of *Language and thought of the child* and *Judgment and reasoning in the child*. *New Ideas in Psychology*, 18(2000), 241-259.
- Ponitz, C. C., McClelland, M. M., Jewkes, A. M., Connor, C. M., Farris, C. L., & Morrison, F. J. (2008). Touch your toes! Developing a direct measure of behavioral regulation in early childhood. *Early Childhood Research Quarterly*, 23, 141–158.
- Ponitz, C. C., McClelland, M. M., Matthews, J. S., & Morrison, F. J. (2009). A structured observation of behavioral regulation and its contributions to kindergarten outcomes. *Developmental Psychology*, 45(3), 605-619.
- Prensky, M. (2001). Digital natives, digital immigrants. *On the Horizon*, 9(5), n.p. NCB University Press.
- Resnick, M. (2006). Computer as paintbrush: Technology, play, and the creative society. In D. Singer, R. Golikoff, & K. Hirsh-Pasek (Eds.), *Play = learning: How play motivates and enhances children's cognitive and social-emotional growth*. New York, NY: Oxford University Press.

- Resnick, M. (2007). Sowing the seeds of a more creative society. *Learning and Leading with Technology*, 35(4), 18 – 22.
- Resnick, M. (2013). Learn to code, code to learn: How programming prepares kids for more than math. EdSurge (May 8, 2013). Retrieved from <https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn>
- Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, 52(11), 60-67.
- Reynolds, A.J., Temple, J.A., Ou, S.R., Arteaga, I.A., and White, B.A.B. (2011). School-based early childhood education and age-28 well-being: Effects by timing, dosage, and subgroups. *Science*, 333(6040), 360-364.
- Roberts, D. F., & Foehr, U. G. (2008). Trends in media use. *The Future of Children*, 18(1), 11–37.
- Rothbart, M.K. (2007). Temperament, development, and personality. *Current Directions in Psychological Science*, 16(4), 207-212.
- Rothbart, M.K., Sheese, B.E., & Posner, M.I. (2007). Executive attention and effortful control: Linking temperament, brain networks, and genes. *Child Development Perspectives*, 1(1), 2-7.
- Rushkoff, D. (2010). *Program or be programmed: Ten commands for a digital age*. New York, NY: O/R Books.
- Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology*, 17, 59 – 69.

- Salomon, G. & Perkins, D.N. (1987). Transfer of cognitive skills from programming: When and how? *Journal of Educational Computing Research*, 3, 149-169.
- Sameroff, A.J. & Haith, M.M. (1996). *The Five to seven year shift: The age of reason and responsibility*. Chicago: University of Chicago Press.
- Scribner, S. (1990). Reflections on a model. *Quarterly Newsletter of the Laboratory of Comparative Human Cognition*, 12(2), 90 – 94.
- Shade D., Nida, R., Lipinski, J., & Watson, J. (1986) The effects of Microcomputers on young children: An examination of free-play choices, sex differences, and social interactions. *Journal of Educational Computing Research*, 2(2), 147-168.
- Shuler, C. (2007). *D is for Digital*. New York, NY: The Joan Ganz Cooney Center at Sesame Workshop.
- Schunk, D. H., & Zimmerman, B. J. (1997). Social origins of self-regulatory competence. *Educational Psychologist*, 32, 195-208.
- Steele, C. M. (1997). A threat in the air: How stereotypes shape intellectual identity and performance. *American Psychologist*, 52, 613–629.
- STEM Smart Brief (2013). Nurturing STEM skills in young learners, PreK–3. Retrieved from <http://successfulstemeducation.org/sites/successfulstemeducation.org/files/STEM%20Smart%20Brief-Early%20Childhood%20Learning.pdf>
- Turkle, S. & Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs*, 16(1), 128-157.

- Vohs, K.D. & Baumeister, R.F. Understanding self-regulation: An Introduction. In R.F. Baumeister & K.D. Vohs (Eds.) *Handbook of Self-Regulation: Research, theory, and applications* (pp. 1-9). New York, NY: Guilford Press.
- Vygotsky, L. (1978). *Mind in Society*. Harvard University Press: Cambridge, MA.
- Wang, X.C., & Ching, C.C. (2003). Social construction of computer experience in a first-grade classroom: Social processes and mediating artifacts. *Early Education and Development, 14*(3), 335 – 361.
- Wanless, S. B., McClelland, M. M., Tominey, S. L., & Acock, A. C. (2011). The influence of demographic risk factors on children’s behavioral regulation in prekindergarten and kindergarten. *Early Education & Development, 22*, 461–488.
- Wartella, E., & Jennings, N. (2000). Children and computers: New technology – Old concerns. *Children and Computer Technology, 10*(2), 31 – 43.
- Wing, J.M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33 – 35.
- Yelland, N. (2005). Mindstorms or a storm in a teacup? A review of research with Logo. *International Journal of Mathematical Education in Science and Technology, 26*(6), 853 – 869.
- Zelazo, P.D., Carter, A., Reznick, J.S., & Frye, D. (1997). Early development of executive function: A problem-solving framework. *Review of General Psychology, 1*, 198-226.

Zillien, N. & Hargittai, E. (2009). Digital distinction: Status-specific internet uses. *Social Science Quarterly*, *90*(2), 274-291.

Table 1

BRIEF - HTKSPre Correlations

BRIEF SCALE	<i>n</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Inhibit	35	33	-0.11	0.51	-.43, .23
Shift	35	33	0	0.99	-.33, .33
Working Memory	35	33	-0.09	0.60	-.41, .25
Plan/Organization	35	33	0.01	0.98	-.32, .34
Initiate	35	33	-0.05	0.78	-.38, .28
Organization of Materials	35	33	-0.01	0.96	-.34, .32
Emotional Control	35	33	0.10	0.58	-.24, .42
Monitor	35	33	-0.04	0.81	-.37, .30

Table 2

BRIEF - HTKSPost Correlations

BRIEF SCALE	<i>N</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Inhibit	38	36	-0.19	0.24	-.48, .14
Shift	38	36	0.00	0.96	-.32, .32
Working Memory	38	36	-0.08	0.65	-.39, .25
Plan/Organization	38	36	0.01	0.96	-.31, .33
Initiate	38	36	-0.02	0.91	-.34, .30
Organization of Materials	38	36	0.01	0.96	-.31, .33
Emotional Control	38	36	-0.02	0.89	-.34, .30
Monitor	38	36	-0.04	0.80	-.36, .28

Table 3

BRIEF – Programming Score Correlations

Correlation	<i>n</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Inhibit	37	35	-0.26	0.12	-.54, .07
Shift	37	35	0.12	0.48	-.21, .43
Working Memory	37	35	-0.05	0.76	-.37, .28
Plan/Organization	37	35	0.02	0.89	-.31, .34
Initiate	37	35	-0.03	0.87	-.35, .30
Organization of Materials	37	35	-0.03	0.88	-.35, .30
Emotional Control	37	35	0.00	0.99	-.32, .32
Monitor	37	35	-0.03	0.85	-.35, .30
GLOBAL	37	35	-0.04	0.8	-.36, .29

Table 4

BRIEF – Time on Task Score Correlations

Correlation	<i>n</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Inhibit	36	34	-0.17	0.32	-.47, .17
Shift	36	34	0.01	0.97	-.32, .37
Working Memory	36	34	-0.13	0.46	-.43, .21
Plan/Organization	36	34	0.04	0.82	-.30, .36
Initiate	36	34	-0.12	0.48	-.43, .22
Organization of Materials	36	34	0.05	0.77	-.28, .37
Emotional Control	36	34	-0.06	0.73	-.38, .27
Monitor	36	34	-0.02	0.91	-.35, .31
GLOBAL	36	34	-0.06	0.74	-.38, .27

Table 5

BRIEF – Goal Completion Score Correlations

Correlation	<i>n</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Inhibit	37	35	-0.15	0.39	-.45, .18
Shift	37	35	0.08	0.64	-.25, .39
Working Memory	37	35	-0.02	0.89	-.34, .31
Plan/Organization	37	35	0.09	0.59	-.24, .40
Initiate	37	35	0.04	0.81	-.29, .36
Organization of Materials	37	35	0.11	0.51	-.22, .42
Emotional Control	37	35	-0.02	0.91	-.34, .31
Monitor	37	35	0.09	0.61	-.24, .40
GLOBAL	37	35	0.02	0.93	-.31, .34

Table 6

Self-Regulation – Programming Score Correlations

Lesson	<i>N</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Airplane (<i>M</i> = 2.62, <i>SD</i> = 1.27)	27	25	0.14	0.49	-.25, .49
Race (<i>M</i> = 3.63, <i>SD</i> = 1.38)	28	26	0.04	0.83	-.34, .41
Sun (<i>M</i> = 3.13, <i>SD</i> = 1.26)	29	27	0.20	0.83	-.18, .52
Dance (<i>M</i> = 2.92, <i>SD</i> = .98)	23	21	0.11	0.61	-.31, .50
Greet (<i>M</i> = 2.52, <i>SD</i> = .58)	25	23	0.14	0.50	-.27, .51
Average (<i>M</i> = 2.95, <i>SD</i> = .79)	35	33	0.14	0.41	-.20, .46

Table 7

Initial Self-Regulation - Goal Completion Score Lesson Correlations

Lesson	<i>n</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Airplane (<i>M</i> = .90, <i>SD</i> = .67)	27	25	0.13	0.52	-.26, .49
Race (<i>M</i> = 1.45, <i>SD</i> = .63)	27	25	-0.05	0.81	-.42, .34
Sun (<i>M</i> = 1.32, <i>SD</i> = .75)	29	27	0.47	0.01*	.12, .71
Dance (<i>M</i> = 1.04, <i>SD</i> = .53)	23	21	-0.23	0.29	-.59, .20
Greet (<i>M</i> = 1.19, <i>SD</i> = .68)	25	23	0.23	0.26	-.18, .58
Project (<i>M</i> = 1.19, <i>SD</i> = .67)	33	31	0.25	0.16	-.10, .55
Average (<i>M</i> = 1.19, <i>SD</i> = .37)	34	32	0.36	0.03*	.03, .63

Table 8

Self-Regulation – Time on Task Score Correlations

Lesson	<i>n</i>	<i>df</i>	<i>r</i>	<i>p</i>	95% CI
Airplane (<i>M</i> = .34, <i>SD</i> = .23)	28	26	-0.11	0.57	-.47, .27
Race (<i>M</i> = .60, <i>SD</i> = .23)	27	25	0.02	0.94	-.37, .39
Project (<i>M</i> = .39, <i>SD</i> = .28)	33	31	0.29	0.10	-.06, .58
Average (<i>M</i> = .43, <i>SD</i> = .17)	34	32	0.23	0.2	-.12, .53



Figure 1. ScratchJr Main Screen. A screenshot of the main screen of the ScratchJr programming language.

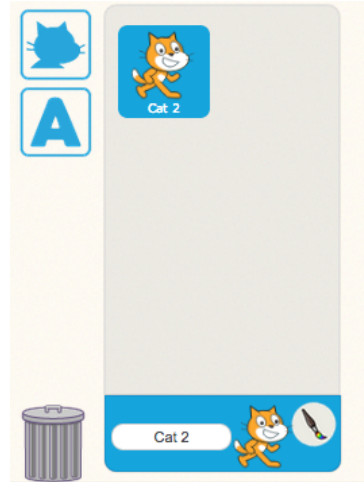


Figure 2. Character and Text Selection. A user chooses the cat icon to select a character, the “A” icon to add text, the trash can to delete a character, and the paintbrush to color the selected character. The highlighted box around the character shows which of the characters is selected for programming.

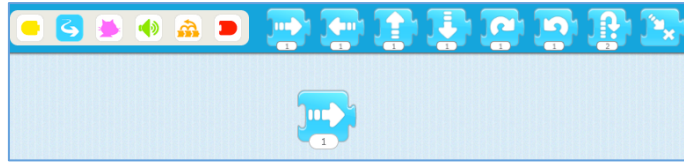


Figure 3. Motion Blocks. The motion icon is selected from the palette on the left to display all the motion block options on the right. A motion block is then dragged to the scripting area below to assign it to the character.

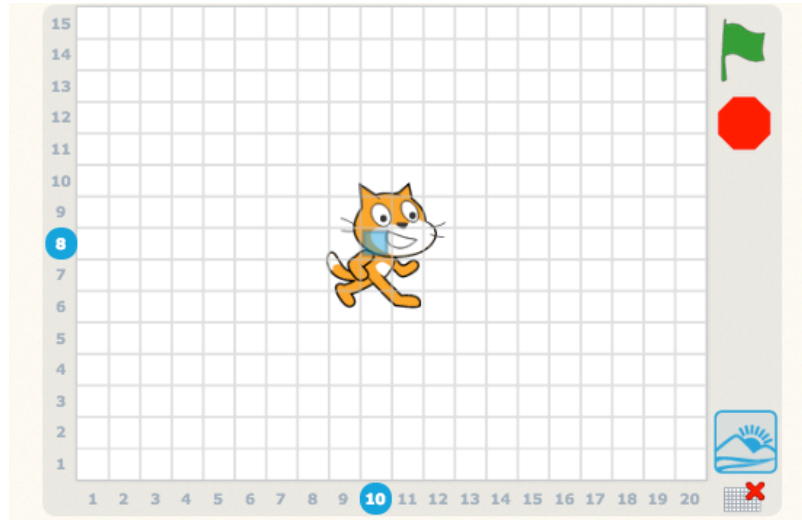


Figure 4. Grid Feature. The character's location is marked on the x- and y-axes. The lower right hand corner displays an option for turning the grid feature on and off. Directly above, the mountain icon allows for selection of a new background. The flag and stop sign start and stop the action on screen.



Figure 5. Lesson Three. A screenshot of ScratchJr, Lesson 3, where pig, caterpillar, and chicken are programmed to race. The “Start on Flag,” “Set Speed,” “Move Forward,” and “End” blocks are shown.

Classroom	Pre-Test	Time 1	Mid-Point	Time 2	Post-Test
Pre16 (Intervention)	HTKS	8 Lessons of ScratchJr		8 Lessons of ScratchJr	HTKS BRIEF (by teachers)
Pre8 (Intervention)	HTKS	8 Lessons of alternative technology tool (KidPix)		8 Lessons of ScratchJr	HTKS BRIEF (by teachers)

Figure 6. Procedure Timeline (75 days total).



Figure 7. Camtasia Screen. This is a screenshot of ScratchJr with Camtasia video recording turned on. Note video of child on right.

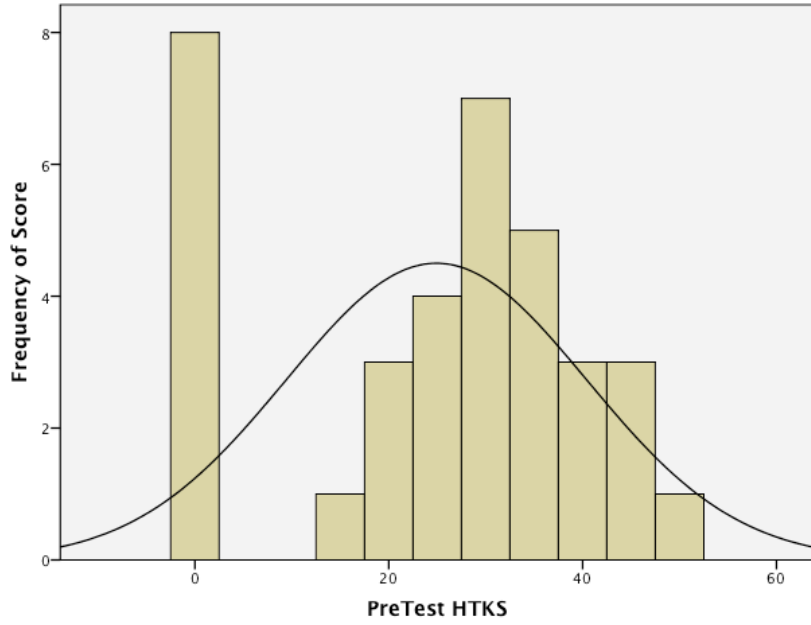


Figure 8. Distribution of HTKS Pre-Test Scores.

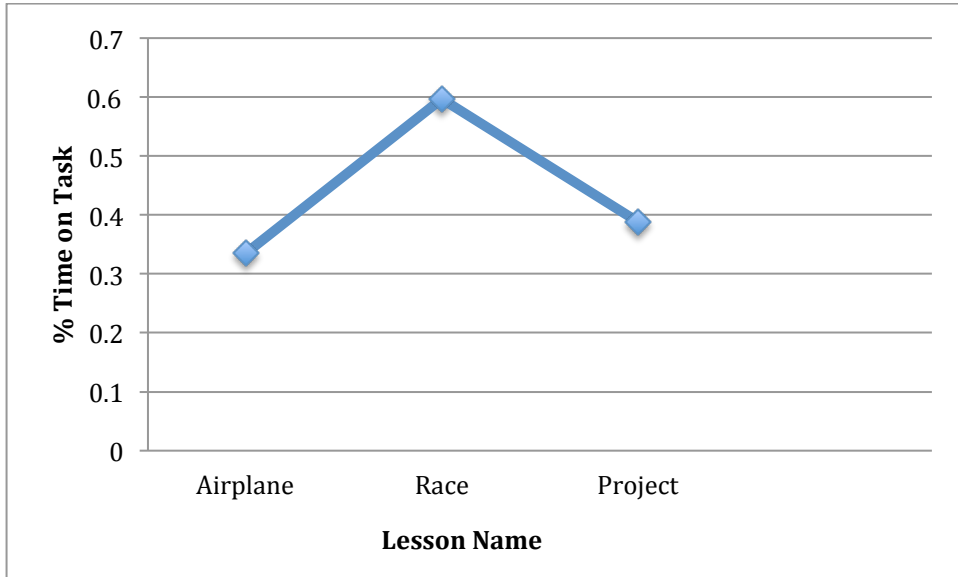


Figure 9. Time on Task Percentage by Lesson.

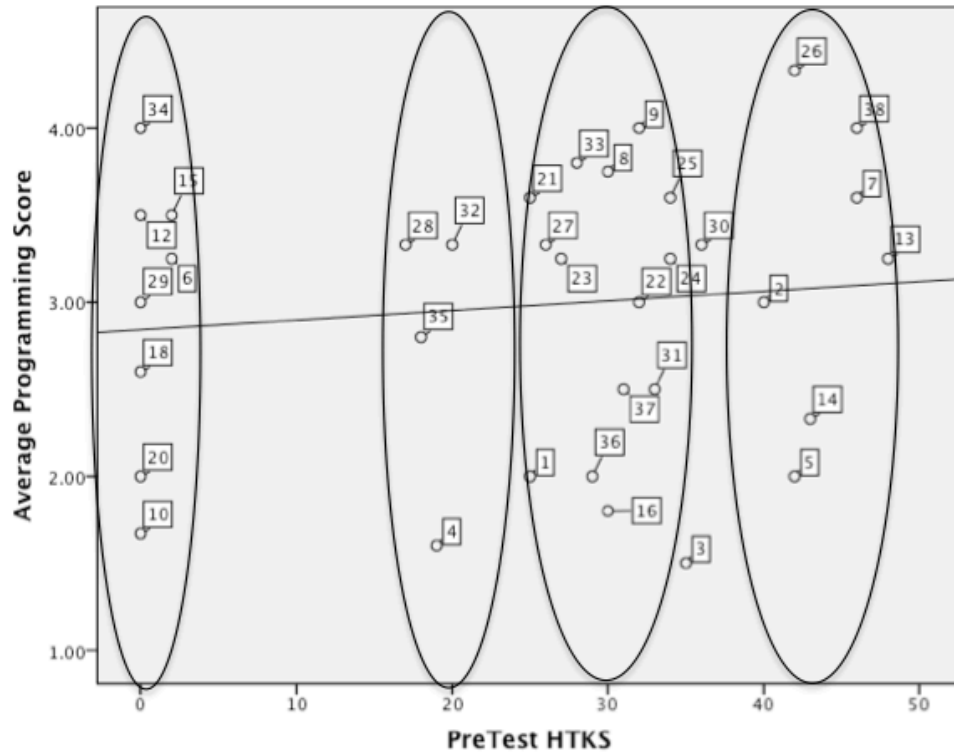


Figure 10. Programming Score and Self-Regulation Scatterplot.

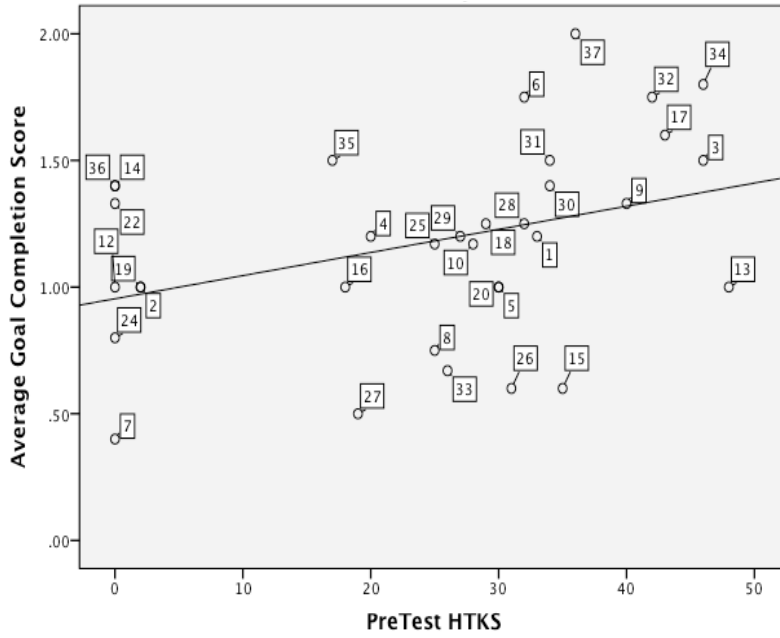


Figure 11. Goal Completion Score and Self-Regulation Scatterplot with Projects.

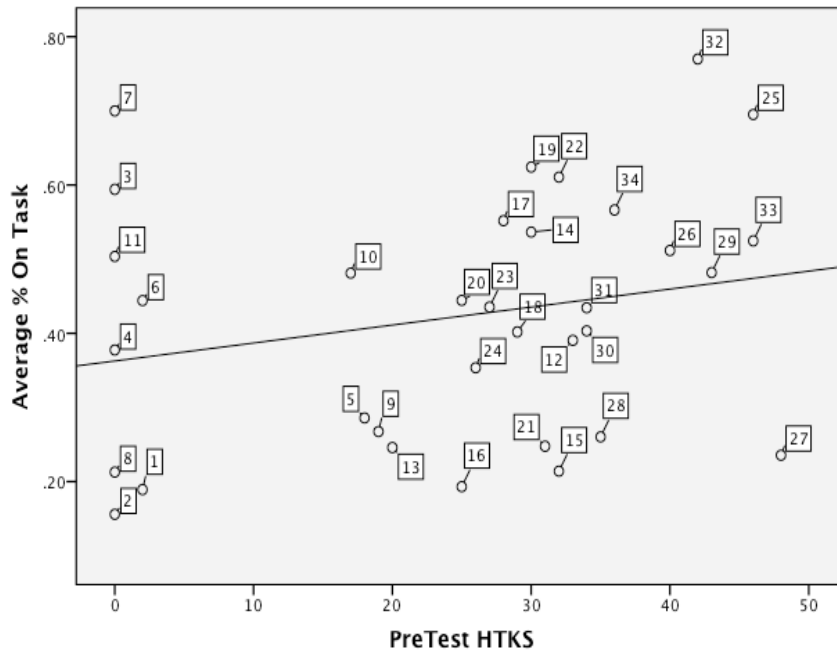


Figure 12. Average Time on Task Percentage and Self-Regulation.

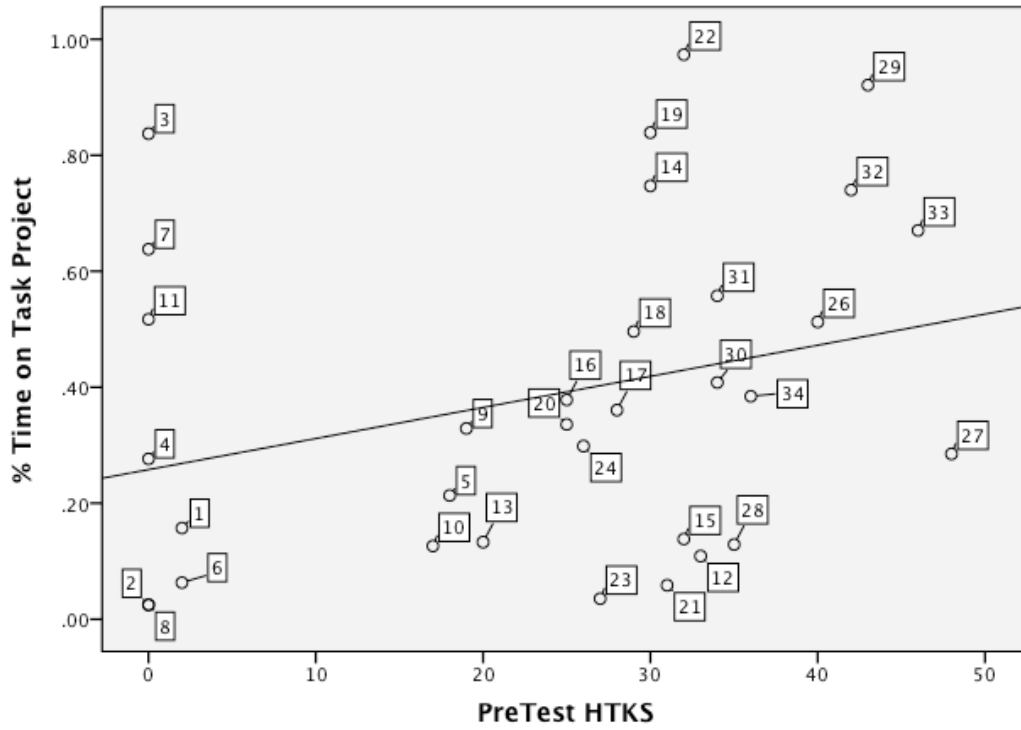


Figure 13. Average Time on Task Percentage and Self-Regulation for Projects.

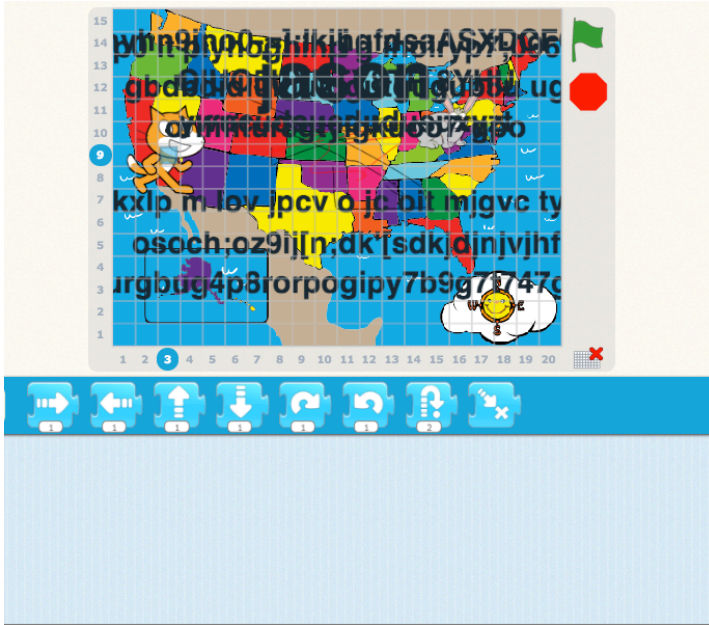


Figure 14. Jack's Airplane Lesson Scratch Jr Project. *About here.]* Jack primarily colored or experimented with adding text. Notice there is no coding in the scripting area.



Figure 15. The End State of Sarah's Race Program.

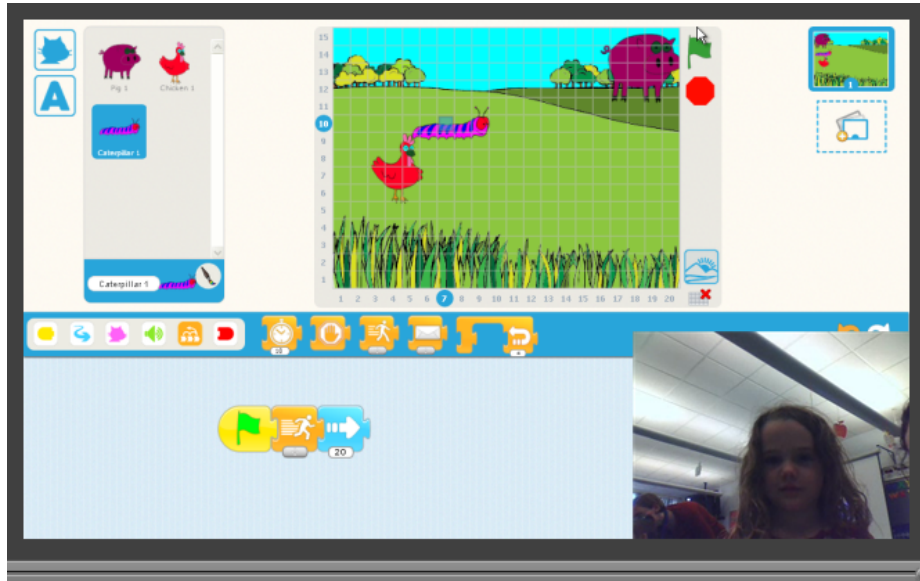


Figure 16. Screenshot of Sarah's Race Project with Colored Characters.

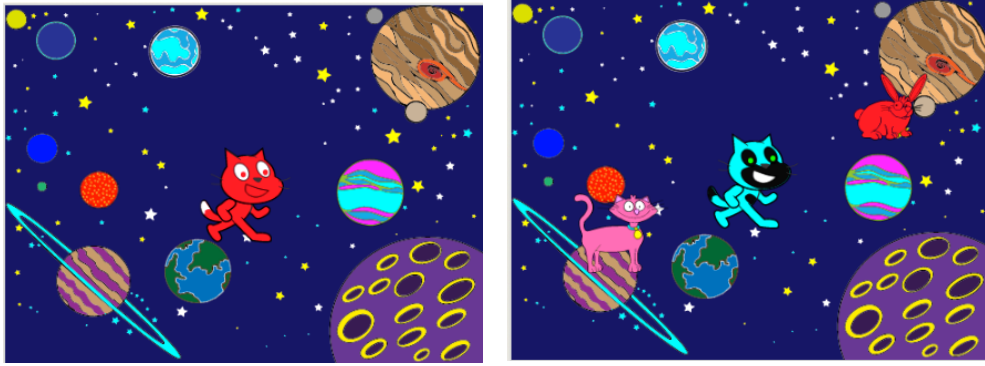


Figure 17. Examples of Cats in Space ScratchJr programs.



Figure 18. Parallel Programming. Eight cars, an airplane, and the girl all have the same program. The number parameter was changed to 20 for each. The sun is decorative (no code attached). The effect is a street full of moving traffic and airplane flying overhead when the program is run.