Code and Tell:

An Exploration of Peer Interviews and Computational Thinking With ScratchJr in the Early Childhood Classroom

A thesis submitted by

Dylan J. Portelance

In partial fulfillment of the requirements of

Master of Arts

in

Child Study and Human Development

TUFTS UNIVERSITY

April 2015

ADVISER: Marina Umaschi Bers

Abstract

The ScratchJr programming language and iPad app were created in a collaboration by Tufts University's Developmental Technologies Research Group, MIT Media Lab's Lifelong Kindergarten Group and the Playful Invention Company in order to provide young children with a developmentally appropriate way to learn foundational computer programming while creating animated stories and games. In this thesis, I present a novel activity called "Code and Tell," designed specifically to supplement early childhood classrooms learning to code and create projects with ScratchJr. Students in three second grade classes learned foundational computational thinking concepts using ScratchJr and applied what they learned to the creation of animated collages, stories, and games. They participated in the Code and Tell activity three times, which involved conducting artifact-based video interviews with each other in pairs using their iPad cameras. Through an exploration of the Code and Tell activity, this thesis seeks to begin to understand the computational thinking learning opportunities that ScratchJr provides to the early childhood classroom when combined with a peer interviewing activity.

Acknowledgments

To my adviser, Professor Marina Umaschi Bers, thank you for guiding me from the playpen to the playground, a lesson I will always keep with me. To my thesis readers, Professor Michelle Wilkerson-Jerde, and Professor Bruce Johnson, thank you for your time, energy, patience, and brilliance. To the always dynamic and compassionate DevTech family, Amanda Sullivan, Elizabeth Kazakoff, Claire Caine, Melissa Lee, and Mollie Elkin, thank you for a never-ending two year stream of new ideas, sharp critique, and unwavering support. A special thank you to two DevTechers—Louise Flannery, for leading me into my first early childhood classroom (since my own childhood), and Amanda Strawhacker for always going above and beyond to help me as I began to lead my own studies. Thank you to the ScratchJr product team for letting me in on decades of learning technologies experience. Thank you to the incredibly sharp and creative crew of undergraduates I was fortunate to enlist—Alex Pugnali, Andre Newland, Elizabeth Dielentheis, Emily Naito, Megan Souza, Sarah Hogan, and Yesenia Villanueva-Rodriguez, I was lucky to have your help and look forward to all of your bright futures. Thank you to the Eliot-Pearson community for being an unlikely home to someone looking for a more personally meaningful way to use computer science. Of course, this work would not have been possible without the students, teachers, staff, families who graciously provided me with their time and space to teach, facilitate, observe, and learn. Thank you. Finally, to my friends and family here and at home who are undoubtedly tired of hearing me talk about children using ScratchJr, thank you for your tireless love and support throughout this process.

Table of Contents

List of Figures

List of Tables

1. Introduction

Motivation

"Learn to code" is more than a suggestion from your principal. "Learn to code" is a movement, an industry, a policy, a course, a promise, and even a value. Through its recent entrance into the national political, educational, and technological spotlights, the phrase has gained influence, garnered an enormous amount of attention, and taken on a life of its own. In 2014, United States President Barack Obama wrote his highly publicized first line of Javascript and became one of over 100 million people worldwide to have participated in Code.org's Hour of Code event. In 2013, New York City Mayor Bill de Blasio launched the Tech Talent Pipeline, aiming to give hundreds of after school programs access to free computer science learning materials from Google. Demographically minded organizations like Black Girls Code, founded in 2011, and Girls Who Code, established in 2012, have offered a programming education to thousands of students from groups usually underrepresented in computer science classrooms and technology businesses. Companies like Codecademy, created in 2011, and Wonder Workshop, whose first investment came in 2013, have raised millions of dollars from venture capitalists with promises to help the masses learn to code.

Why all this attention toward writing instructions for a computer to read and execute? Learning to code ostensibly grants people access to job opportunities within rapidly growing markets (Landivar, 2013) as well as a gateway to participation in digital artistic expression and civic engagement (Jenkins et al., 2006). An argument for learning to code that is particularly popular among computer science scholars and technology creators is the epistemological benefit. Alan Perlis, recipient of the first Association for Computing Machinery A.M. Turing Award, suggested that the skill be "part of every liberal education," contending that the deep

understanding of process could be transferrable to calculus, economics, and many other domains (Perlis, 1962). Seymour Papert claimed more generally that computer programming propagates the opportunity to think about one's own thinking (1980). As programmers continually build more complex systems, they in turn reflect on deeper complexities of their own personal thinking processes.

The ubiquity of computing devices and software in our everyday lives makes learning to code seem even more imperative. We engage with digital technology as we send messages on phones, purchase goods with Amazon.com, gather knowledge from Wikipedia, sign petitions on Change.org, and fulfill an endless variety of other personal and collective purposes. For those coming of age in an increasingly technological world, developing an understanding of how these technologies work under the hood may help them navigate this environment with greater ease or even build technological solutions of their own. Media theorist Douglas Rushkoff went so far as to dub coding "the new literacy," with many others following suit, arguing that the ability to program digital media has reached an equal level of importance to the ability to read and write print media (2010).

This theme of new literacy closely accompanies ScratchJr, the technology used during the classroom studies for this thesis. ScratchJr is an animation-making app designed to provide young children with a developmentally appropriate way to learn to code. With the public release of ScratchJr in August 2014 came the tagline, "Coding is the new literacy!" and an explanation that with ScratchJr, "children aren't just learning to code, they are coding to learn" (ScratchJr.org). This position implies that the practice of coding, like reading and writing, can facilitate the development of other skills like problem solving and design as well as build knowledge in other domains such as mathematics and language. As a product made to create

these kinds of learning opportunities for young children and given the potential affordances of these learning opportunities, children's use of ScratchJr brings about many questions. What exactly are young children learning as they create projects with ScratchJr? Are they just learning to code, or are they engaging with other powerful ideas from computer science? How can we know what they have learned? What kind of literacy are they developing?

To begin answering these questions, this thesis focuses on learning of the general skill set known as *computational thinking*, to which coding belongs. Computational thinking, first coined by Papert, describes not only the processes and concepts used to solve problems and design systems with computers but also the application of ideas inspired by computers that humans can use to aid in expressing and understanding the nature of phenomena (1996). The term was most exhaustively defined by Jeannette Wing, who explains it as a rich set of analytical methods used to effectively combine human and machine toward solving problems (2006). These methods include more concrete tasks like programming, testing, and debugging as well as abstract ideas like decomposition and representation of data.

Thesis Outline

In this thesis, I explore the use of a curricular activity that I call "Code and Tell." The activity consists of pairs of students conducting video interviews with each other about ScratchJr projects that they created during class. The research question that this thesis seeks to answer is: How can the Code and Tell activity be used to provide students with opportunities to learn computational thinking in the early childhood classroom? First, I review the literature on technological tools for learning computational thinking as well as learning and assessing computational thinking. Second, I discuss the primary technology used during the research, the ScratchJr iPad app. Third, I describe the research design, including a curriculum enactment and

data collection methods. Fourth, I analyze the data to address the aforementioned research question. Fifth, I address some of the limitations of the present study. Sixth, I offer implications of this thesis research as well as future directions.

2. Background

Computing is more common than ever in education, industry, and life in general, and the impetus for society to have a better understanding of how computing works has never been stronger. Recently, a great plurality of the computer science education community has converged to meet this need by pushing for students to learn a skill set called computational thinking. For scholars and practitioners in computer science, education, and computer science education, the scope of this push has involved theoretical contributions about the ontology of computational thinking, views on how to implement computational thinking activities for educational purposes, and studies on how computational thinking is actually learned. For the purposes of this thesis, I will summarize our current understanding of computational thinking and then focus on literature that provides a foundation for my own work regarding methodology for assessing the learning of computational thinking as well as specific applications of these assessments for use with young children.

What is Computational Thinking?

Defining computational thinking is a hot topic for computer science education scholars and multiple explorations of the construct have been put forth. To recapitulate the definitions offered in the Introduction, Papert is credited as the first to employ the term but he did not rigorously define it (1996). As he uses the term in his writing, he signals a mode of thought that could be harnessed toward problem solving and system design with computers in concurrence with using ideas drawn from how computers operate to help see and interpret processes, ideas,

and rules. In her short paper entitled, “Computational Thinking,” Jeannette Wing illustrates the term’s breadth in detail, citing a vast array of computer science related skills and concepts such as abstraction, decomposition, and gauging time and space costs (2006). Like Papert, she emphasizes the term’s simultaneous reference to techniques enacted by machines and by humans.

Brennan and Resnick break down computational thinking more specifically into a three-dimensional framework that comprises concepts, practices, and perspectives (2012). Computational thinking concepts are the most concrete of the bunch, individual operations of computation that can be interpreted and executed by a machine and are oftentimes built into a programming environment as a feature. Examples of computational thinking concepts include sequence—a set of instructions meant to be executed in a given order—and data—information that can be stored, outputted, mutated and interpreted to determine some computation. At a higher level, computational thinking practices refer to techniques applied by humans to design and construct computations. Some of these computational thinking practices are debugging, procedures and strategies for hunting down, understanding, and fixing computation so that it does not cause unwanted behavior, and abstraction, the hiding of unnecessarily complex details of a system to facilitate simpler interaction with it. Existing at the highest level of the three dimensions are computational thinking perspectives. While concepts are embedded in the tool and process of computation, and practices reside in the work of creating computation, computational thinking perspectives encompass overarching purposes for computing. Connecting to other people and expressing ideas or identity are some examples of these perspectives.

Computational Thinking and Education

Despite some disagreement about the meaning of the construct, a strong consensus exists that computational thinking ought to be treated as a crucial learning objective for K-12 students. After Wing published her seminal description of computational thinking, many people in the computer science education community turned their attention to creating definitions and frameworks that could be useful in educational settings. In 2010, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) led a National Science Foundation project entitled, "Leveraging Thought Leadership for Computational Thinking in PK-12." One of the sponsored activities of this project was a meeting of education minds who set out to construct a definition of computational thinking as it should be interpreted and applied in school curriculum. In addition to summarizing many of the skills introduced by Wing, this "operational definition" included a list of attitudes toward dealing with computing problems such as confidence in the face of complexity and ambiguity and working with others communicatively and collaboratively (International Society for Technology in Education and Computer Science Teachers Association, 2011).

Building on the claims that computational thinking can be applied to multiple domains of knowledge and practice for students, many have attempted to show examples of what this could look like (Barr & Stephenson, 2011; Dierbach et al., 2011). For example, using abstraction in language arts could be likened to employing figurative language such as simile or metaphor, and parallelization has relevance in science because experiments are often run simultaneously with different parameters. Others have investigated ways to integrate computational thinking into interdisciplinary after-school activities (Lee et al., 2011). Research on specific computational learning environments has also shown the existence of different types of learners (Turkle &

Papert, 1990) and probed the motivational potential of different computing interfaces for engaged student participation (Repenning et al., 2010).

Methodology for Evaluating Learning of Computational Thinking

The computational thinking literature is rife with theoretical contributions and justifications for the creation of technologies and pedagogy that build computational thinking skills. Very recently, research that also investigates the efficacy of existing programs and practices for learning computational thinking has grown more common. Only a few years ago, the Association for Computing Machinery (ACM) and CSTA released a report claiming that computer science assessments are, "virtually non-existent" (Wilson et al., 2010). Since then, however, computer science education researchers have frequently constructed and experimented with more concrete ways to evaluate student learning.

One approach to understanding student learning of computational thinking is by analyzing students' projects within some programming environment. Han Koh et al. (2010) extracted "computational thinking patterns" from thousands of middle school students' game projects based on their use of different programming language constructs and then analyzed how often patterns used in these projects were transferred to use in science simulation projects. Stolee and Fristoe (2011) ran a similar analysis on Kodu projects created by students to see the frequency with which different components of the Kodu language were applied to their projects, including Boolean logic, control flow, objects, and variables. Also in this vain, Denner, et al. (2012) looked at content of games created by middle school girls in Stagecast Creator to assess their understanding of programming and usability. To construct a programming project analysis specifically for primary grades, Seiter and Foreman (2013) proposed the Progression of Early Computational Thinking (PECT) Model, which takes into account students' use of more concrete

evidence variables like loops and conditionals, design patterns like user interaction and animation, and more high level computational thinking concepts like decomposition and abstraction. These studies acknowledge that although mapping the learning of computational thinking concepts directly from the use of different language constructs is convenient for analyzing large data sets automatically, what is gained in efficiency may be lost in validity. Students' use of computer science concepts does not guarantee that they have learned them.

Triangulating student artifacts like programming projects with other data about student learning can more thoroughly evaluate learning trajectories and outcomes. To assess middle school students' learning of computational thinking concepts throughout a computer science summer camp that used Scratch for lessons and projects, Franklin et al. (2013) combined assessment of smooth execution and robust implementation in students' programming projects with fields notes documenting the levels of help that were provided to those students by the camp staff. Basawapatna et al. (2011) supplemented student project data with scores on a quiz meant to measure whether students could transfer use of computational thinking patterns in a programming context to recognition of those patterns in non-programming contexts.

In a few studies that focused on active student performance within an assessment environment, students were evaluated on their abilities to construct programs to solve new problems, debug solutions to these kinds of problems, and demonstrate their aptitude for recognizing and comprehending elements of a programming language. In 2012, Werner et al. developed a computational thinking assessment, using the Alice programming software, that asked students to solve problems by writing code related to characters in a narrative on the screen. To fulfill a similar purpose in a tangible programming domain, Fields et al. (2012) presented students with non-functional circuits in order to assess their ability to debug them by

constructing new functional versions and using computational thinking skills related to circuit design in the process.

Interviews between teachers and students have been used to elicit computational thinking concepts in speech. Dwyer et al. (2014) analyzed discussions with themes of computer knowledge, complex decisions, and sequential procedures to understand fourth graders' development of sequence and algorithm implementation abilities in a CS Unplugged curriculum. Brennan and Resnick (2012) established a framework for assessing computational thinking that is threefold, including elements of some of the previously described techniques. This framework involves analysis of a portfolio of students' projects, interviews with students' projects as a central focus, and "design scenarios," activities that allow students to show what they know in the context of a specific problem situated in the programming environment they have been learning and creating projects with.

The Case of Young Children

It is less common to see computational thinking learning environments and pedagogy designed specifically for young children and assessments that are developmentally appropriate for this demographic are even more rare. Studies have shown that even children as young as four years old can use simple programming interfaces to create robotics projects (Bers et al., 2002; Cejka et al., 2006; Kazakoff et al., 2012; Perlman, 1976; Wyeth, 2008) and animation projects (Strawhacker & Bers, 2014; Portelance et al., 2014). These endeavors help young learners engage with powerful ideas from technology, including many computational thinking concepts, that can serve them in educational and personal pursuits throughout their lives (Bers, 2008).

With particular regards to early childhood computational thinking assessments, Bers et al. (2014) evaluated kindergarten students on their sequencing and instruction recognition

capabilities by scoring their programs written in a tangible wooden block programming language for the purpose of designing a robot dance with specific steps. Strawhacker and Bers (2014) applied assessments to young children's learning of ScratchJr by evaluating their ability to recognize and place programming blocks in a sequence corresponding to the implementations of animations created by researchers.

Jean Piaget's seminal theory of cognitive development, which includes several qualitative stages of development that outline how children conceptualize the world, may be useful to consider while designing computational thinking assessments for young children. According to Piaget, children enter the preoperational stage of development at around two years of age and enter the concrete operational stage at approximately their seventh year of age. During the preoperational stage, children show an ability to understand, manipulate, and play with symbols as stand-ins for real things (1929). The preoperational stage is also characterized by egocentrism, or difficulty with understanding the perspectives of other people, and the development of a curiosity and frequent question-asking behavior (1929). As children transition from the preoperational stage to the concrete operational stage, they begin to develop logical and conservational thought and their egocentrism diminishes qualitatively (1929).

Lev Vygotsky's zone of proximal development may also be useful to consider while designing computational thinking assessments for young children. Vygotsky suggested that every child has a developmental level in a given domain (e.g. computational thinking) that they reach through their own thinking as well as the capacity to develop within that domain to another level with the guidance of others (1978). He emphasized that the way in which children could develop mental processes with the help of others involves the more developmentally mature members of a cultural group interacting with the less mature members through language and that

these linguistic interactions could help children create shared meanings in a given cultural context (1978). With specific regards to peer collaboration and the zone of proximal development, Tudge (1992) notes that interactions between two peers do not necessarily involve a more competent peer guiding a less competent peer within his or her zone of proximal development. He proposes that mutual student interest and verifiable results of development are just some of the factors involved in making these collaborations more effective (1992).

To provide a framework for how technologies can support young people's positive development, Bers (2006; 2012) offers the Positive Technological Development (PTD) framework. Drawing from Lerner et al.'s (2005) *Six C's of Positive Youth Development*, PTD suggests that young people engaging with technology can do so in a way that promotes positive developmental outcomes when technology substantiates activities revolving around communication, collaboration, community-building, content creation, creativity, and positive choices of conduct. This framework has been applied to the creation of developmentally appropriate robotics kits and programming environments for young children as well as playful and technology-rich activities for young children (Bers, 2010).

Gap in the Literature

If ScratchJr is to be used as a tool for learning computational thinking in the early childhood classroom then there exists a need for a developmentally appropriate computational thinking assessment that can be used with students learning in this environment. Ideally, this assessment can simultaneously serve as an activity that supports positive technological development as well as assessing. The present study seeks to address this gap in the literature by investigating how the Code and Tell activity can be used to provide students with opportunities to learn computational thinking in the early childhood classroom.

3. ScratchJr

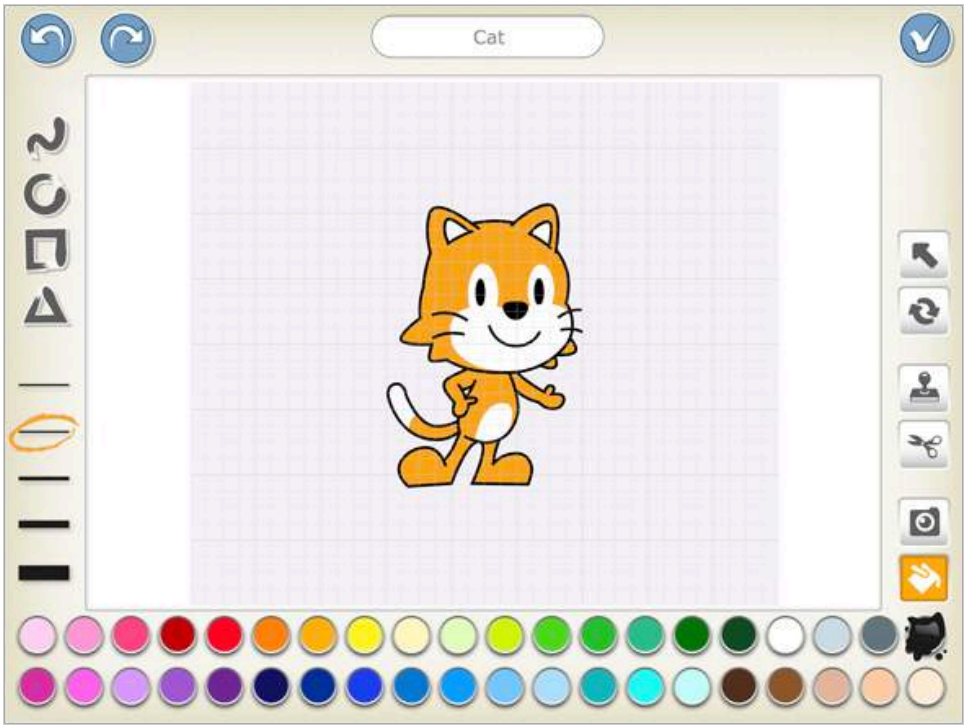
ScratchJr was created to provide young children with developmentally appropriate means to learn to use computer programming skills as well as to make personally meaningful projects with technology (Flannery et al., 2013). The three major components of the ScratchJr project are: 1) the app, which comprises a programming language and interactive animation-making interface designed specifically for use by young children, 2) curricular resources and content within the app that allow for integration with early childhood mathematics and literacy learning, and 3) online resources for early childhood educators to learn about ScratchJr content, activities, and teaching practices. Throughout the app's design and development, researchers conducted curriculum enactments in K-2 classrooms focusing on powerful ideas related to computer science and technology (2013). These curricula served simultaneously as ways to test the app itself and to iterate on its design as well as to assess activities and best practices for teaching with ScratchJr in an early childhood classroom setting.

App Features

At the crux of the ScratchJr iPad app is a graphical programming language consisting of 28 programming blocks. Users connect these blocks together in sequences for the purpose of programming characters on the screen to move, change their appearances, produce sounds, and utilize a variety of other functions towards the creation of animated stories and games on the iPad. Other features supplement the results made possible by ScratchJr's programming blocks. With the ScratchJr interface, users can further adapt their project by adding multiple pages, adding characters and backgrounds from the libraries, creating their own characters or backgrounds using the paint editor, adding text to the page, or utilizing other parts of the technology.

Figure 1 – ScratchJr Programming Interface

Figure 2 – ScratchJr Paint Editor



Design

ScratchJr was based on the online programming language and environment, Scratch, created for children ages 8 and older, but unlike Scratch, ScratchJr was designed to be developmentally appropriate for young children, specifically ages 5-7. To center developmental appropriateness for young children during ScratchJr's development, several features and design decisions construct a "low floor" for users (2013). It is easy for users to begin programming within seconds because when the app is opened to the project interface, there is already a Cat character on the stage and the blue motion blocks are immediately visible and available to program with. Users can simply drag and drop any of the eight motion blocks into the scripting area below the blocks palette and tap them with their finger to see the result of a simple one-block motion program. To encourage users to create programs with more than one block, the blocks are shaped like jigsaw puzzle pieces that obviously snap together. Syntax errors are impossible in the ScratchJr programming language because blocks that can only be placed at the beginning of a program are rounded on the left side and blocks that belong at the end of a program are rounded on the right side. Additionally, users can drag characters around on the stage in order to position them without having to write programs.

The design of the ScratchJr programming language, app, activities, and resources builds upon a large body of research on technological tools for children to learn computer programming with. This work spans studies that examine designing programming technology for children, teaching children how to code, and assessments of children's programming knowledge and skills.

4. Research Design

A study was designed and conducted to address the research question: How can the Code and Tell activity be used to provide students with opportunities to learn computational thinking in the early childhood classroom? The study's design drew from a methodology known as design-based research, which involves testing theories about how learning can take place by enacting iteratively designed interventions in a natural environment (Brown, 1992; Collins, 1992, Cobb et al., 2003). A developmentally appropriate curriculum for learning computational thinking with ScratchJr was designed and taught to a selection of second grade students. As part of the curriculum intervention, each of the students participated in the Code and Tell artifact-based video interviewing activity multiple times. Data was then collected in the form of videos recorded during the Code and Tell activity and field notes taken by researchers in the classroom.

Sandoval's Conjecture Map

Sandoval's (2014) conjecture mapping approach, a technique created for the purpose of "conceptualizing design-based research" and structuring it with an "argumentative grammar," was used to guide the method and analysis. Conjecture maps begin with a high-level *conjecture* about how learning is supported without specifics about the designed learning environment. They also include an *embodiment,* or learning environment design made up of tools, participant structures, task structures, and settings. Next, conjecture maps involve *mediating processes,* or links between the learning environment design and desired outcomes, verifiable through artifacts created by learners or observations of interactions between learners. Finally, conjecture maps include *outcomes*, or manifestations of successful learning. Table 1 shows a conjecture map that outlines the high-level conjecture, embodiment, mediating processes, and outcomes that comprise the focus of the present study.

Table 1 – Conjecture Map Outlining the Structure of the Present Study

<table>
<tr><th>Conjecture</th><th>Embodiment</th><th>Mediating Process</th><th>Outcome</th></tr>
<tr><td rowspan="3">The Code and Tell activity can support the learning of computational thinking with ScratchJr in the early childhood classroom</td><td>Tools
• ScratchJr iPad app
• Camera iPad app</td><td rowspan="3">Use of computational thinking beyond concepts represented as features in ScratchJr iPad app</td><td rowspan="3">Development of computational thinking ability</td></tr>
<tr><td>Activity Structure
• Designing and constructing personally meaningful ScratchJr Animated Genre projects
• Presenting ScratchJr projects to peers and iPad camera during Code and Tell activity
• Interviewing peers about ScratchJr projects using iPad camera during Code and Tell activity</td></tr>
<tr><td>Participant Structure
• Participation in ScratchJr Animated Genres curriculum taught by ScratchJr tutors with assistance from regular classroom teachers</td></tr>
</table>

The present study seeks to address how the Code and Tell activity provides students opportunities to learn computational thinking. While the ultimate goal of effective use of the Code and Tell activity in early childhood classrooms learning ScratchJr is the outcome stated above, development of computational thinking ability, evaluating whether the learning design achieves this outcome is beyond the scope of this study. This study will focus on a mediating process that is hypothesized to potentially lead students situated in the described embodiment to the desired outcome. This mediating process is the use of computational thinking beyond concepts represented as features in the ScratchJr iPad app. Without loss of generality, examples of concepts represented as features in the ScratchJr iPad app are loops (tangibly represented as the Repeat block) and sequencing (represented as the ability to connect programming blocks together), whereas examples of computational thinking practices that are not represented as physical features in ScratchJr are debugging and iterative design. Specifically, analysis of the videos that students record during their Code and Tell activity participation will be used to inform a description of the nature of this mediating process, if it emerges from the learning design in the first place.

Setting

The study was conducted in three second grade classrooms at a suburban public elementary school in the Greater Boston area. This school was selected for three reasons. First, there was a professional connection between its principal and Tufts University's Eliot-Pearson Department of Child Study and Human Development. Second, the school had recently acquired about thirty iPads through a generous grant. This number of iPads made it possible for each classroom to distribute them in such a way that each student would have their own for the duration of any given lesson period. Finally, the school was located in close enough proximity to

Tufts University to make it possible for researchers to travel to and from the research site without conflicting with other commitments on campus.

Materials

While all three classrooms were set up differently due to the preferences of the regular classroom teachers, there were several features of the classrooms that were consistent. The following features particularly characterized the setting and affected the way activities would be conducted throughout the curriculum.

- iPad cart

  The three classrooms shared one set of iPads, meaning each student shared their iPad with two other students, each in a different classroom. These iPads would arrive in the appropriate classroom just before the ScratchJr tutors arrived to facilitate the day's activities. On most days, there would be a few minutes at the beginning and end of the lesson period dedicated to distributing iPads to their respective users and putting the iPads back in the cart.

- Whiteboards

  Each classroom had several whiteboards on the walls. These whiteboards would be used to display an enumeration of the day's activities or to provide a reference for the interview questions used during the Code and Tell activity.

- Student tables

  Students in all three classes had assigned seats at tables distributed throughout their classrooms. These tables usually seated about four to five students. Generally, while students were using the ScratchJr iPad app, they were seated in their assigned seats at these tables but this was loosely enforced.

- Rug area

  A rug with enough space to seat an entire class of students served as a place for gathering everyone closely in the classrooms. When introducing new concepts in ScratchJr, tutors would have students be seated at the rug area without their iPads. Students would also be gathered at the rug area while waiting for their turn to participate in the Code and Tell activity.

- Overhead projector

  All of the classrooms had overhead projectors, which ordinarily could be used to display handouts, manipulatives, and other classroom materials at a large scale. This technology was used sparingly by ScratchJr tutors after the first few lessons due to the perceived lack of student engagement when using the device.

Participants

Sixty-six students in the second grade participated in the curriculum enactment. During lesson periods, regular classroom teachers were present at all times. With few exceptions, two ScratchJr tutors, including the thesis author, were present in the classroom leading the activities. Although these tutors facilitated, regular classroom teachers were encouraged to “take the wheel” whenever they felt comfortable. The ScratchJr tutors also let these teachers know that because of their knowledge of their classroom’s culture and best practices for classroom management that they could make modifications to the plan for a lesson period in order to make it as appropriate as possible for their students. Other adults were sometimes present in the enactment classrooms. These included paraeducators, who would sometimes visit to provide special assistance to individual students and student teachers from another university shadowing the regular classroom teachers.

Curriculum

The curriculum taught during the study comprised a 13-day one-hour-per-lesson program with lessons occurring twice a week. The ScratchJr "Animated Genres" curriculum allows students to familiarize themselves with and use all of the different computational and creative aspects of the ScratchJr iPad app. Simultaneously, it seeks to help students engage with "powerful ideas" from computational thinking, like debugging and iterative design, that can be applied to projects and thinking in other domains such as writing or science (Papert, 1980).

Split into three modules focusing on three animated genres of communication, students repeatedly engaged with two lessons on utilizing features in ScratchJr, a lesson period dedicated to working on personal projects within an animated genre, and a lesson period for participating in the Code and Tell activity. After the third time through the Code and Tell activity, families were invited into the classroom for a "Family Day." Students showcased their projects for family and friends to present what they had learned and created during the curriculum enactment. Table 2 provides an outline of the curriculum.

Table 2 – ScratchJr Animated Genres Curriculum Outline

Day	Module	Activity
1	Create a ScratchJr Collage	Learn about Motion blocks
2		Learn about Looks blocks, "Start on Green Flag" block, Characters, and Backgrounds
3		Make ScratchJr Collage
4		Code and Tell with Collage projects
5	Create a ScratchJr Story	Learn about Control blocks
6		Learn about Text, Pages, Sound blocks, End blocks
7		Make ScratchJr Story

8		Code and Tell with Story projects
9	Create a ScratchJr Game	Learn about “Send Message” block and “Start on Message” block
10		Learn about “Start on Tap” block and “Start on Bump” block
11		Make ScratchJr Game
12		Code and Tell with Game projects
13	Family Day	Show family and friends ScratchJr collages, stories, and games

Animated Genre Projects

The lesson periods on days three, seven, and eleven gave students the entire allotted time to focus on designing and constructing their own personal ScratchJr projects. Projects would be composed to fit the animated genre that the current module focused on. ScratchJr tutors introduced each animated genre project by reflecting on what students had learned earlier in the module and discussing the elements of that genre (e.g. a story has characters, setting, beginning, middle, and end).

Animated Collage Project

On the third day of the curriculum enactment, students were given a full lesson period of one hour to work on a ScratchJr collage project. The collage project was introduced with a brief discussion about collages made without iPads. ScratchJr tutors emphasized how regular collages on paper combined many different elements on one page and how students could make collages in ScratchJr but with the added benefit of animation. They were encouraged to use the features they had learned in ScratchJr so far, including the motion and looks blocks, the “Start on Green

Flag" trigger block, the character library, the background library, and the paint editor. Figure 3 shows an example of a ScratchJr animated collage project.

Figure 3 – Example of a ScratchJr animated collage project

Animated Story Project

On the seventh day of the curriculum, students were given a full lesson period of one hour to work on a ScratchJr story project. The story project was introduced with a brief discussion about stories without iPads. ScratchJr tutors emphasized how regular stories on paper used different pages with writing and pictures about characters in a setting to combine a beginning, middle, and end into a plot. They were encouraged to use the features they had learned in ScratchJr so far with a special emphasis on the features they learned between the Collage Project day and the Story Project day, the control blocks, text editor, sound blocks, and end blocks. Figure 4 shows an example of a ScratchJr animated story project.

Figure 4 – Example of a ScratchJr animated story project

Animated Game Project

On the eleventh day of the curriculum, students were given a full lesson period of one hour to work on ScratchJr game projects. The game project was introduced with a brief discussion about games without iPads. ScratchJr tutors emphasized in the discussion that games have rules and objectives. They also emphasized students building ScratchJr game projects would need to design a way for players to interact with them. They were encouraged to use programming blocks and features that they learned between the Story project day and the Game project day, especially the “Start on Tap” block, which is required for building an interactive ScratchJr project. Figure 5 shows an example of a ScratchJr animated game project.

Figure 5 – Example of a ScratchJr animated game project

Code and Tell Activity

Each project lesson period was followed immediately by a lesson period dedicating to participation in the Code and Tell activity (days four, eight, and twelve). Regular classroom teachers partnered students based on their previous success working together as reading, writing, or mathematics partners. Students remained with the same partners throughout the entire study with only a few rare but necessary exceptions due to absences and other unexpected circumstances.

In the classroom, the interviews were announced as a way for students to present their ScratchJr projects to classmates using their own words. The ScratchJr tutors also reiterated this purpose prior to the second and third comings of the activity on days eight and twelve. The interview questions were written on a whiteboard at the beginning of a lesson period dedicated to interviews and were read aloud by the ScratchJr tutors to the participants a moment before they

began recording. Table 3 delineates the Code and Tell interview questions, with the last question being a question of the interviewer's choice.

Table 3 – Code and Tell Interview Questions

1	Tell me about your project.
2	How did you make your project?
3	What would you do if you had more time?
4	Your choice.

Data Collection

For those students with consent, forty-two total students, videos recorded during the Code and Tell activity were collected and analyzed to address the research question: How does the Code and Tell activity provide students' opportunities to learn computational thinking in the early childhood classroom? Due to absences, six students only completed two interviews, meaning the total number of students whose full set of three video interviews were analyzed was thirty-six. Results of the analysis and a discussion of the findings are provided in the following chapter.

5. Results and Discussion

The purpose of the present study is to address the following research question: How does the Code and Tell activity provide students' opportunities to learn computational thinking in the early childhood classroom? In order to address this question, sixty-six students in three second grade classrooms participated in a ScratchJr Animated Genres curriculum enactment where they each took part in the Code and Tell activity up to three times. The videos that students filmed of each other during their participation in the Code and Tell activities were collected for analysis if

the student appearing in the video had consent. Videos from a total of forty-two students were analyzed and, due to absences, full sets of three videos from thirty-six students were analyzed. The other six students with consent only completed the activity twice had two videos. A qualitative data analysis (Miles et al., 2014) was conducted on these videos with the aim of providing descriptions of the development of computational thinking in an early childhood classroom using the Code and Tell activity. A qualitative analytical approach was chosen because the research question is an exploratory one attempting to understand the behavior of students in a new activity setting. Since the activity is new, a conceptual framework that does not take into account findings within the qualitative data itself would be based on a limited range of theory and knowledge.

Categories of Analysis

In order to build a conceptual framework allowing one to analyze the Code and Tell video data set, holistic coding (Jones, 1985; Dey, 1993) was conducted on the videos to extract major categories of what students talk about and do during the Code and Tell activity. These categories were gleaned based on "general comprehension of the data" (Dey, 1993) with the intention to break them down into subcategories based on complexity levels of computational thinking. There were three categories that emerged that encompass students' behavior while sharing their ScratchJr projects: *describing projects, demonstrating projects, imagining projects.* These themes are defined in Table 4.

Table 4 – Code and Tell Video Categories of Analysis

Category of Analysis	**Definition**
Describing Projects	Providing an account of what the project is.

Demonstrating Projects	Displaying the project in a way that provides observable information that can used to better understand it.
Imagining Projects	Conceiving what the project could be and strategizing about what will constitute it.

Connections to Computational Thinking

Each of the categories that emerged from initial analysis—describing, demonstrating, and imagining projects—has a strong connection to computational thinking. These connections make each category a fruitful area for more in depth exploration. The following explanations illustrate their connections to computational thinking.

Describing Projects

Being able to think about what something is without enumerating all of its components and details is necessary for thinking computationally. This practice allows computer scientists to represent information, processes, and interfaces in ways that render them manageable and understandable to themselves and others. When students engage in the act of describing a ScratchJr project, they talk about an idea or multiple ideas of what that project is. Whether consciously or not, they select a level of abstraction at which to represent their project and build a description at that level. In the context of a student talking about a ScratchJr project of their own design, these descriptions could manifest itself in several ways. Students may list the characters in a project, share its title, dive into the plot of a story, or speak on behalf of the intent behind its creation. These descriptions may reflect how students conceptually represent the project.

Demonstrating Projects

Problems in computer science often require a solution in the form of a demonstrable product. For example, if you need to know the measures of central tendency on a large data set, an equation is not enough. What would be more useful would be a program, because a program can be executed to produce understandable output that can then be usefully leveraged. The practice of demonstrating a computational entity involves ensuring its readiness, familiarizing oneself with its functionality, and taking into account a viewer's prior understanding of that entity. When students demonstrate their ScratchJr project during Code and Tell, they engage in the practice of sharing information with their audience in order to effectively evoke an understanding of that project's function or essence. Many behaviors caught on video comprise the array of demonstration related actions, including explaining what is happening on the screen, repeating an animation, and manually maneuvering characters into new locations with a finger. Demonstrations reflect the student's capacity to understand and share their project.

Imagining Projects

When effective computer scientists build creative solutions to problems, they propose a hypothetical lack of constraints, think across a set of possibilities, and paint pictures of the ideal solution. These imaginative processes sometimes lead to more thoughtful designs and implementations that suit the problem. Students who imagine projects illustrate their intentions when they set out to make a project. They speak about the possibilities of what could be included in their project or what could be happening beyond what the project explicitly shows. A student is imagining his or her project as he or she mentions a full range of things a character could be doing despite the fact that an audience may only see a cat moving forward. Another student imagining their project might have many elements added but has plenty of ideas for how it can

be expanded, even if these additions or not necessarily feasible. Students who imagine present their projects at the intersection of what their creative minds see and the projects themselves show.

Levels of Describing, Demonstrating, and Imagining Projects

Although examples of students describing, demonstrating, and imagining projects were common across the selection of interview videos, there were different ways that students went about doing so. Within each of these three categories, three sub-categories emerged that could classify the level of complexity with which students described, demonstrated, or imagined their projects. Table 5 displays an overview of simple, intermediate, and complex ways of describing, demonstrating, and imagining projects.

Table 5 – Levels of complexity for describing, demonstrating, and imagining projects

	Simple	**Intermediate**	**Complex**
Describing Projects	Explain project elements are	Explain things project elements do	Explain what project elements are for
Demonstrating Projects	Showing individual project elements	Showing project elements as a system	Showing project elements as an interactive system
Imagining Projects	Focus on physical project elements	Focus on conceptual project elements	Focus on audience/user experience

Levels of Describing Projects

Simple Describing

Students describing projects in a simple way talk about what different elements of their projects are. The following excerpt from a Code and Tell interview video provides an example of a student describing a project in a simple way.

Excerpt 1 – Describing a project in a simple way

[0:00] Interviewer: Tell me about your project.

[0:01] Presenter: Well my project is there's a little polar bear and…it's…neon or something…and…um, and it's in Af- I mean Antarctica.

Figure 6 – Scene from interview containing Excerpt 1

In Excerpt 1, the presenter describes her project by listing two elements that she added to it—a polar bear character and an arctic background. She also mentions the new coat of paint she gave her polar bear as well as its size, which she programmed using a "Shrink" programming block. As she describes her project to the interviewer and camera, she homes in on observable characteristics of the elements she added to her project rather than how they act or what they mean.

Intermediate Describing

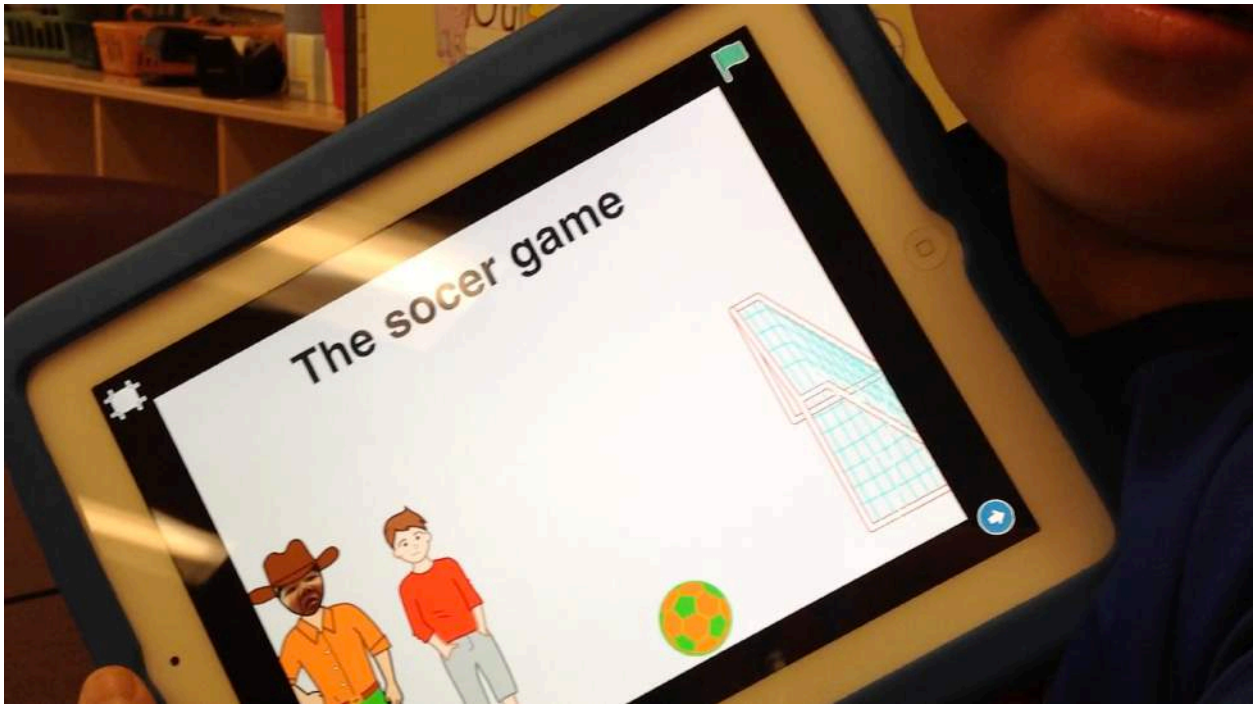
Students who describing projects in an intermediate way talk about what different elements of their projects do. The following excerpt from a Code and Tell interview video provides an example of a student doing intermediate describing.

Excerpt 2 – Describing a project in an intermediate way

[0:01] Interviewer: Tell me about your project.

[0:03] Presenter: Uh, my project, is like, this crazy cowboy, and this teenager, playing soccer…

Figure 7 – Scene from interview containing Excerpt 2



In Excerpt 2, the presenter describes his project by first describing the two characters he chose and customized for his project, and then by describing the activity that he programmed them to take part in. To program this activity he used a series of motion blocks to move the teenage character to look like he is kicking the soccer ball character, another series of motion blocks to move the soccer ball into a goal character, and a “Say” block to program the “crazy

cowboy" to yell, "Gggggggggaaaaaaaa!!!!!" The presenter's description includes what his project elements physically do rather than just what they are, but it does not delve into their individual functions.

Complex Describing Projects

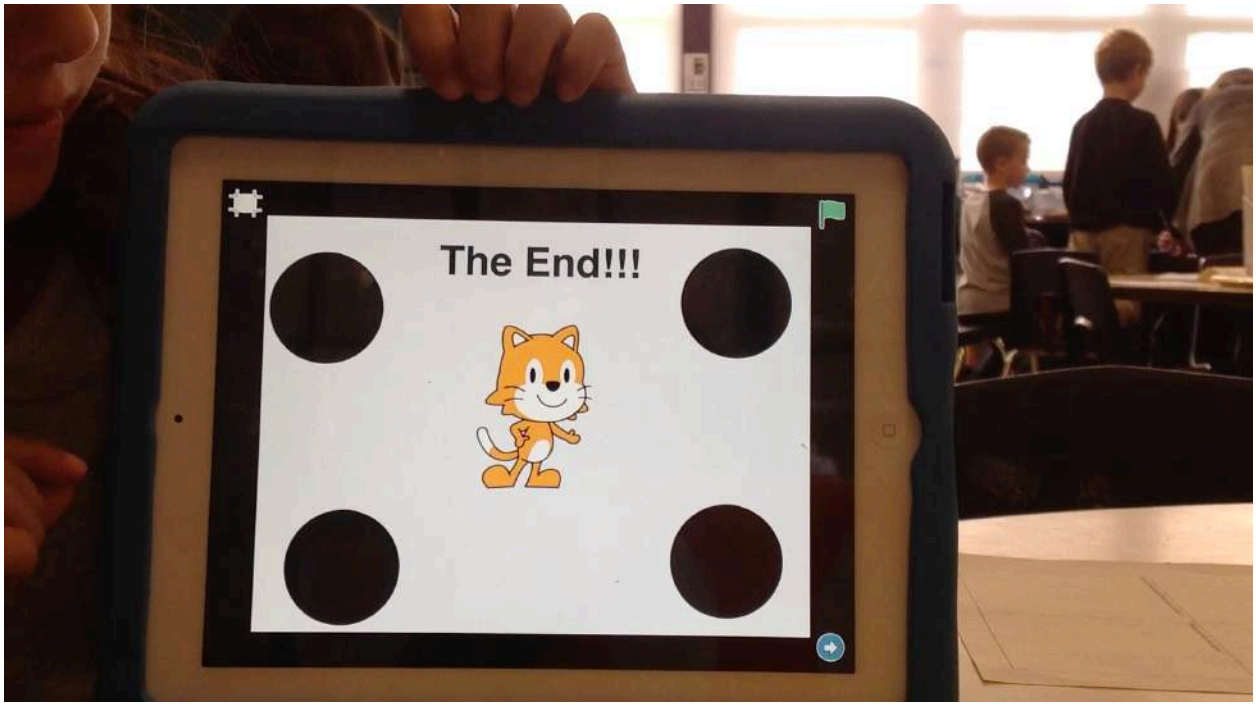
Students describing projects in a complex way talk about the functions of the different elements of their projects. The following excerpt from a Code and Tell interview video provides an example of a student describing a project in a complex way.

Excerpt 3 – Describing a project in a complex way

[0:00] Interviewer: Tell me about your project.

[0:03] Presenter: Um…well, basically what you do in this game, er project, you would press buttons and it would tell you if you won because there would be, it would take you to this background that says you won. Another thing is, if you press the other buttons, it's kind of cool because you press one of these buttons and it will make a new button.

Figure 8 – Scene from interview containing Excerpt 3



In Excerpt 3, the presenter describes a project that comprises several customized "Button" characters. These buttons have been programmed by her to carry out a variety of different functions. At least one button transitions the game player to a new page in the project, yielding a win state in the game. Other buttons ostensibly create new buttons, which lead to more possible options for the game player. In her description, the presenter accounts for the role that her different project elements play in her project rather than just how they act on the screen or what they are.

Levels of Demonstrating Projects

Simple Demonstrating Projects

Students demonstrating projects in a simple way show their peers project elements individually. The following excerpt from a Code and Tell interview video provides an example of a student demonstrating a sequence of steps he took to utilize a feature in ScratchJr towards

the creation of a single element of his project, exemplifying project demonstration in a simple way.

Excerpt 4 – Demonstrating a project in a simple way

[1:52] Interviewer: How about because you're having trouble with this, [name redacted], why don't you, um, show us something you could make your character say? Why don't you make your character say something?

*[2:01] Presenter: Sure! I just did! *turning iPad around to face camera* I just made him say, "Hi [name redacted]." Look! *taps programmed script* "Hi [name redacted]."*

[2:09] Interviewer: That's great.

[2:10] Presenter: Now, now I'm gonna make my dragon say, "Hi Bob."

[2:11] Interviewer: I like it.

[2:17] Interviewer: Okay.

[2:17] Presenter: Would you like to see how I do it?

[2:19] Interviewer: Sure!

*[2:21] Presenter: Here, so, I just press the "Hi" button. *taps Say block on the text to open a keyboard window* "Hi." And now I click the, click the, *taps the B* "B," and now, *taps the O* "O," and then, *taps the B* "B." B-O-*

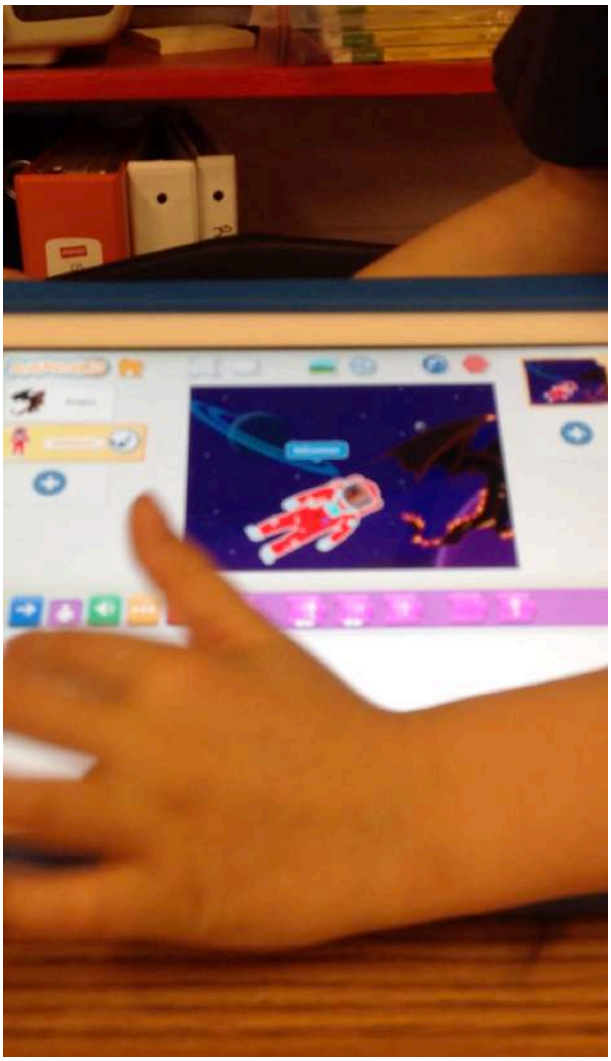
*B, *taps the Go button* do that..., *taps the Dragon's script* "Hi Bob."*

[2:39] Interviewer: And then, what do you press now? Do you press, "Go"?

*[2:40] Presenter: *taps the Dragon's script* "Hi Bob."*

*[2:42] Presenter: Yeah, you just press, "Go." Here, see? Watch, *taps the Dragon's script* "Hi Bob!"*

Figure 9 – Scene from interview containing Excerpt 4



In Excerpt 4, the presenter demonstrates knowledge of how to enter his own parameter into the “Say” block in order to program his character to say something in the form of a speech bubble. Although the interviewer clearly prompts the presenter to demonstrate something, the presenter seems to have already been creating something with the iPad facing away from the camera with the intention of showing it to the interviewer and the camera after he finished. After he does so, he elaborates on how to effectively use a “Say” block by showing another example of what can be programmed to appear in a speech bubble. In his demonstration, the presenter shows how the ScratchJr speech bubble feature works by using the “Say” block and does so within the context of his own project creation. During the excerpt, the presenter demonstrates a single project element in a few ways, typifying simple demonstrating.

Intermediate Demonstrating Projects

Students demonstrating their ScratchJr projects in an intermediate way talk about the different elements of their project as part of an overarching system. The following excerpt from a student’s Code and Tell interview video provides an example of a student demonstrating their project in an intermediate way.

Excerpt 5 – Demonstrating a project in an intermediate way

[0:00] Interviewer: Tell me about your project.

*[0:02] Presenter: Well my stor-, my project’s about a wizard defeating a ranger…on this page, *moves Wizard character around with finger the wizard* the wizard is fighting the ranger. The wizard says, *taps Wizard’s script* “Don’t you get away,” and then *taps the Fairy character button* the fairy doesn’t say anything, *taps the Horse character button* and then the horse doesn’t-, *taps*

the Ranger character button and then the, and then, he said, and then he said, "Yes I will." And then, *taps the second page thumbnail* on this page…um…on this page…um…on this page the wizard doesn't say anything either, and then, um, and then the ranger said, and then…oops *laughs* and then this person says, *taps Ranger character's script* "I'm the best in the world!" *laughs* And then, *taps fourth page thumbanil* at the end, the wizard defeats everybody.*

Figure 10 – Scene from interview containing Excerpt 5

In Excerpt 5, the presenter shows her interviewer and the camera what she put on different pages of her ScratchJr project and how she programmed her characters. The contents of her project form a cohesive narrative about a conflict between different characters. The choice of character types (i.e. wizard, horse, ranger), who appear on which pages out of the four total

pages, their positioning, and their programmed dialogue all combine to tell the story of a wizard "defeating" a ranger. In her demonstration, the presenter begins by talking about the overall theme of the project, the wizard defeating the ranger, and then leaps into what specific components do, each pushing her narrative forward. The presenter demonstrates how project elements that she added, strategically placed, customized, and programmed work together as a system, representing intermediate demonstrating

Complex Demonstrating Projects

Students demonstrating their ScratchJr projects in a complex way talk about the different elements of their project as part of an overarching dynamic system where different things can happen depending on circumstances. The following excerpt from a student's Code and Tell interview video provides an example of a student demonstrating their project in a complex way.

Excerpt 6 – Demonstrating a project in a complex way

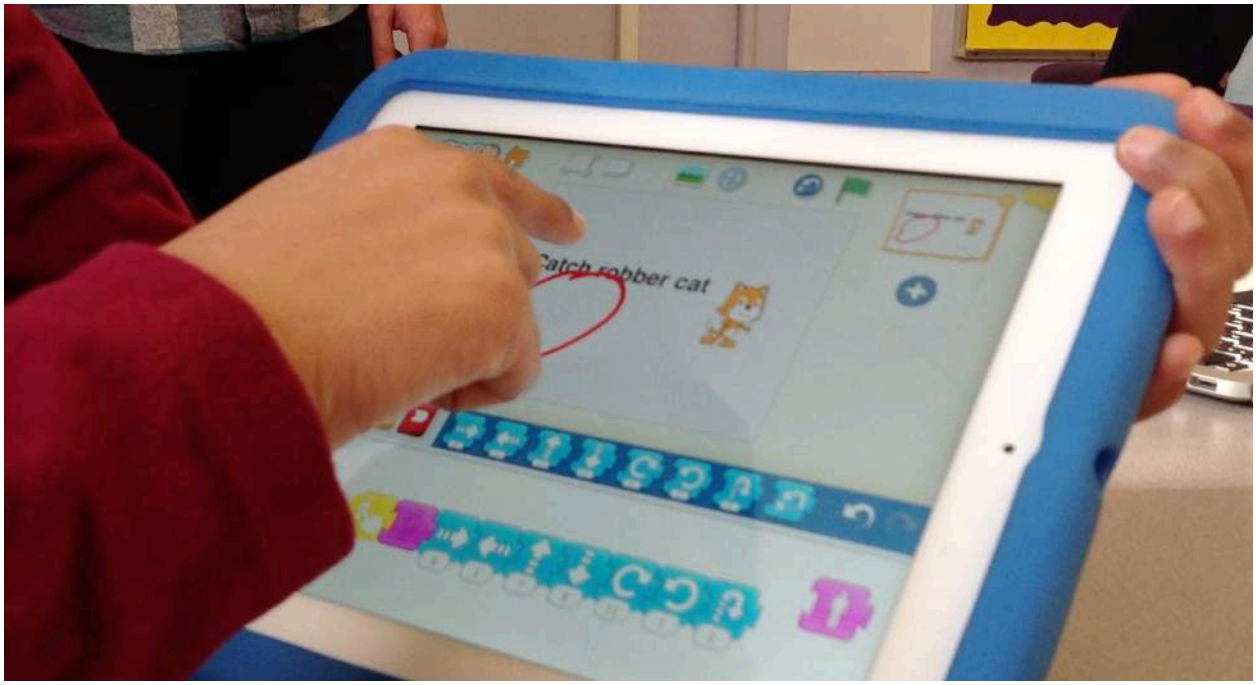
[0:00] Interviewer: Tell me about your project.

*[0:02] Presenter: So it's…you technically *taps the "robber cat" character* touch the robber cat and he disappears and there's also some moves. And then you have to try to put this *drags the red circle character around* on him. But then it would be cheating if you just pressed that *taps Grid button*, 'cause then you know he's right there. *drags the red circle character around quickly and randomly* Hey, where is he? Where is he?*

taps the invisible robber cat's Show block

Wake up.

Figure 11 – Scene from interview containing Excerpt 6



In Excerpt 6, the presenter shows the interviewer a game where you need to move a character customized to look like a red circle around an invisible "robber cat." The presenter demonstrates that in order to start the game, you need to touch the robber cat. At this point, it becomes invisible and then moves to some unknown location. He also demonstrates that if you had touched the grid button, you would know where the invisible cat is because a blue square would be highlighted to show its location. According to the presenter, playing his game with the grid mode on would be cheating for this reason. The presenter then goes on to show an example of how one would play the game if he or she did not know where the cat was (even though it is clear to the presenter, interviewer, and camera where the cat is at this time). Through his demonstrations, the presenter shows elements in his project as part of an interactive system with behavior designed to depend on the user.

Levels of Imagining Projects

Simple Imagining Projects

Students imagining projects in a simple way talk about designing the physical elements of their projects as opposed to the conceptual ones. They also do not focus on the audience or user of their project as they talk about their design. The following excerpt illustrates a student presenting a ScratchJr project and imagining possibilities in a simple way.

Excerpt 7 – Designing a project in a simple way

[0:00] Interviewer: How did you make your project?

[0:04] Presenter: Um, well, it's, it's like the ice scene, and there's, like, a polar bear and a cat, and the cat's supposed to be riding the polar bear.

[0:25] Interviewer: How did you make your project?

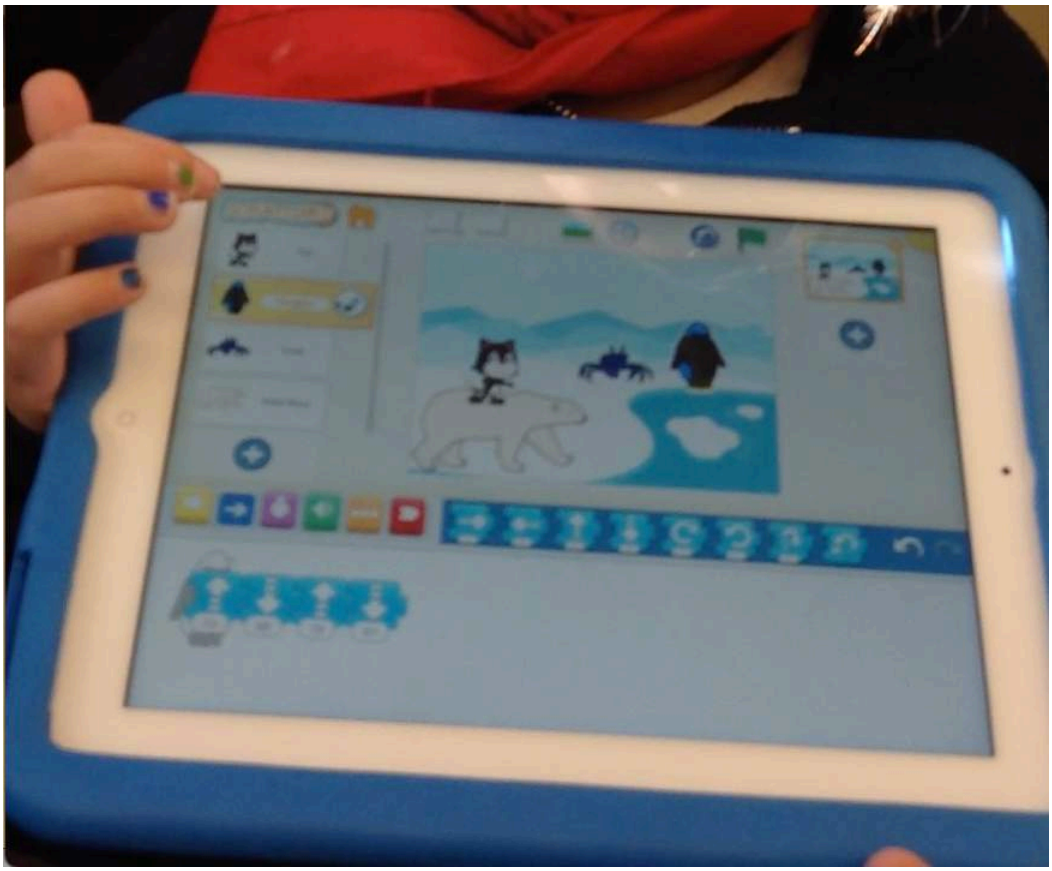
[0:27] Presenter: Well, first I started with the cat and I colored him…and then I thought that a penguin would be good for matching, and I colored him black, blue, and yellow. Then I thought maybe the crab would match so then I made my crab dark blue. And then I made my cat—I thought there should be a polar bear…riding the cat, the cat riding the polar bear, so that's how I did it.

[1:08] Interviewer: Um, what do you do if you had more time?

[1:13] Presenter: I would probably add lots of things that fly

*over here *drags finger along top of screen* and add a fish popping his head *points to the water portion of the arctic background**

Figure 12 – Scene from interview containing Excerpt 7



In Excerpt 7, the presenter touches on a few design decisions she has already made and some that she might make in the future. These design decisions depend on what the project physically looks like. When the interviewer prompts her to talk about her project and asks her how she made it, she focuses primarily on the visible elements she added. She mentions that she has made an ice scene, explaining the arctic background she chose, and then lists the characters that she added, occasionally mentioning details like color. When asked about what she would do with more time to work on her project, she keeps with her theme of animals, saying she would

add things that fly in the background and a fish to the foreground. The presenter exemplifies the simple imagining because of a fixation with physical elements in her project. She asks what other characters could be present that would fit the theme and imagines them alongside to the characters she has already placed on the screen.

Intermediate Imagining Projects

Students imagining projects in an intermediate way focus on conceptual project elements in addition to the physical ones. They may talk about characters, backgrounds, and actions but their focus is on the concept represented by these elements. The following excerpt showcases a student talking about the design of her project in an intermediate way.

Excerpt 8 – Designing a project in an intermediate way

[0:00] Interviewer: Tell me about your story.

[0:01] Presenter: Well, about my project is that they're doing magic tricks on each other and they're performing but no one's there because it's just like practice.

*[0:11] Interviewer: Yeah yeah yeah, I get it. *accidentally covers camera lens with hand* whoa.*

*[0:14] Presenter: So he *points to Cat character on the left side of the screen* says, "Whokey pokey," then he goes small and he disappears, then he *points to Cat character on the right side of the screen* says, "Hilly silly," and he does the same thing. *the script ends and the page is changed automatically, two cats*

appear to be moving from left to right across the screen at different speeds And then, on my next one…*

[0:25] Interviewer: Tell me about, how did you make your project, from that page, the one with the race? How did you use, how did you use the blocks?

*[0:34] Presenter: Well, I wanted them to race in different, like, moves, so I used this one *points to Set Speed block* tell which one I wanted to go fast or slow or medium.*

Figure 13 – Scene from interview containing Excerpt 8

In Excerpt 8, the presenter offers two different scene concepts that she used ScratchJr to realize. The first scene she talks about is a magic show rehearsal. Her description begins with a conceptual idea of what is happening—the characters are performing magic tricks on each other but since there is no audience present, it is only a practice run. She shows how the different elements of her project that she created fit under this theme. The two cat characters say silly things that resemble magic words and they display the after effects of magic tricks like disappearance. Later, she provides another example of a concept in her ScratchJr project—a race between two cats. She communicates to her partner that in order to design characters' motion to fit this concept, she used the "Set Speed" block and set the two cats to different speeds to make the race more meaningful. The presenter exemplifies imagining in an intermediate way because of her ability to support intangible concepts like a magic show and race to what she physically adds to the screen.

Complex Imagining Projects

Students imagining projects in a complex way focus on the experience of the user, player, or audience of their project. They do not just talk about what is happening on the screen or what concepts those physical components of their project fall under. They talk about reaction, mood, and difficulty. They imagine their project under the gaze of a peer, taking into consideration how a viewer might observe, respond to, or use the project. The following excerpt presents a student imagining their project in a complex way.

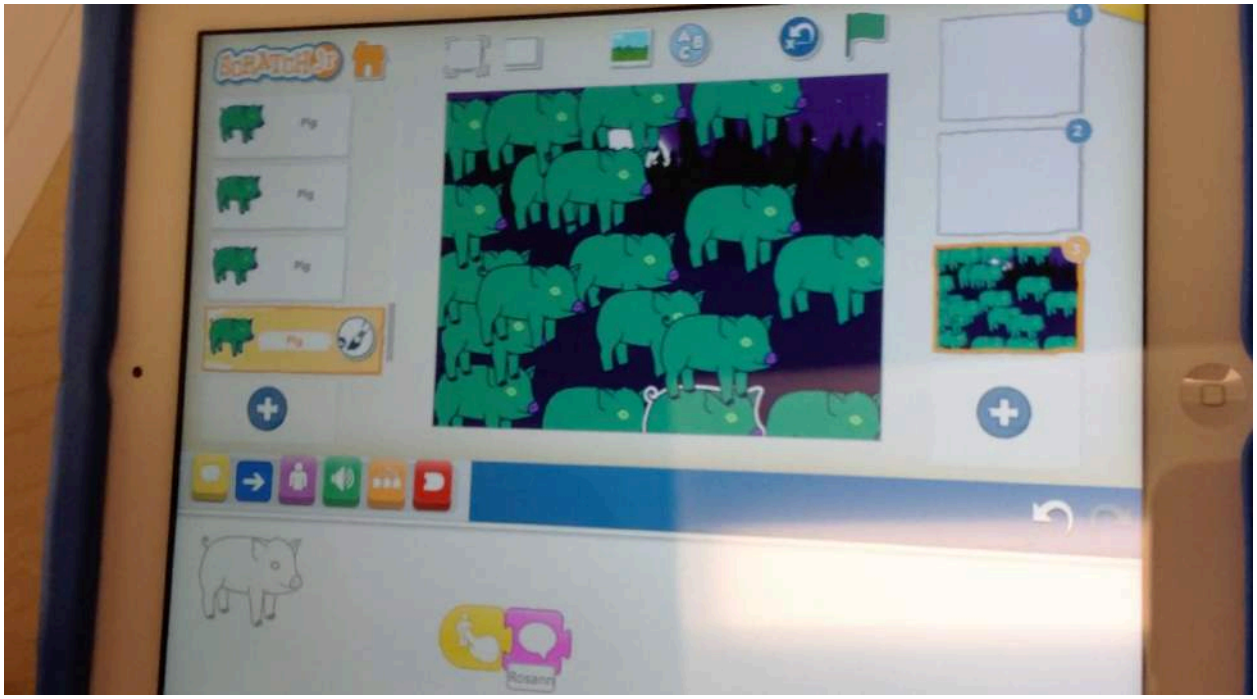
Excerpt 9 – Designing a project in a complex way

[1:45] Interviewer: Tell me about your project.

[1:48] Presenter: Okay, so, like, for example, um, before I let somebody play it I shuffle it and I did a lot of pigs because that would make the

person confused because they're all the same and there's so much pigs.

Figure 14 – Scene from interview containing Excerpt 9



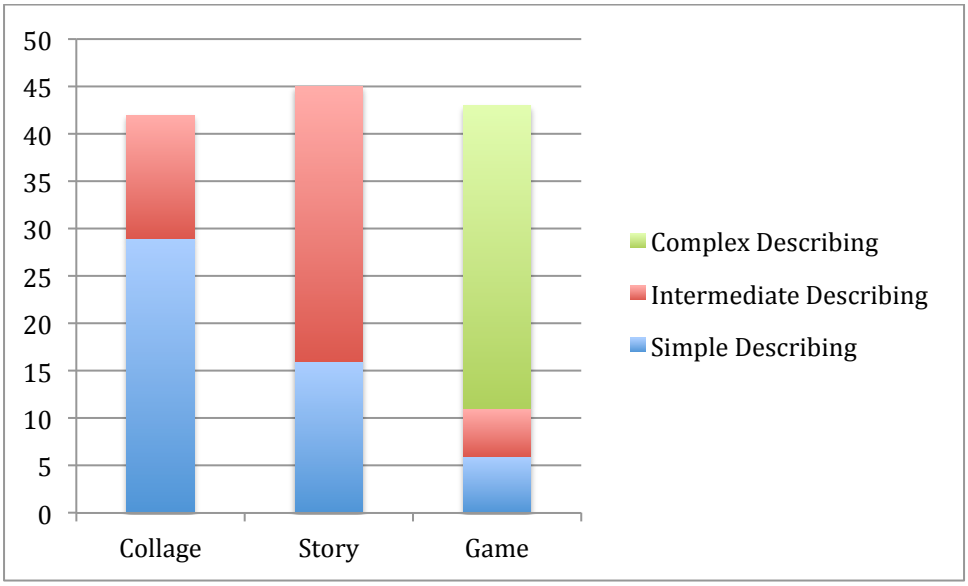
In Excerpt 9, the presenter shows a game project she has made where the player must tap one of the characters on the screen. If the player taps the character that spells her teacher's name correctly in their speech bubble, that player wins the game. When prompted by the interviewer to talk about the project, she talks about how she designed her game based on how she thinks a player would perceive it. She proposes that they would be confused because of the manual shuffling she does each time a player plays and because of the sheer number of characters on the screen. This presenter exemplifies imagining projects in a complex way because she thinks about the potential perspective of the game player she is designing to outsmart with her game's complexity.

Patterns in Whole Group of Students

After categories and sub-categories were discovered during the initial holistic analysis of the collection of Code and Tell interview videos, the videos were coded for these categories and sub-categories. The purpose of this phase of coding was to see if there were any patterns in the ways that the selection of students as a whole moved through these levels of complexity as they participated in the Code and Tell activity at different times during the curriculum enactment. Results from this phase of coding are presented along with a discussion.

Patterns in Levels of Describing Projects

Figure 15 – Frequencies of Levels of Describing Projects Depicted in Videos

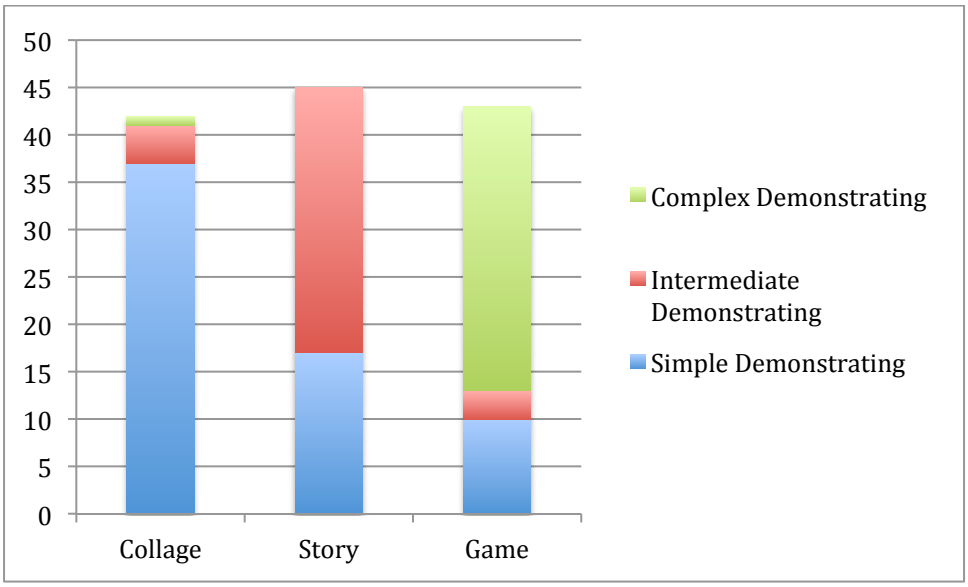


Each of the Code and Tell interview videos was placed in a category based on what level of describing projects it exemplified during that video. Figure 15 shows the overall patterns of frequencies of each level of describing projects as they relate to the interview videos depicting student presenters talking about specific animated genre projects. A look at the chart reveals that students describe projects in a simple manner the most during collage project interviews and this decreases over time. Students describe projects in an intermediate manner the most during the

story project interviews and students describe projects in a complex manner the most during game interviews.

Patterns in Levels of Demonstrating Projects

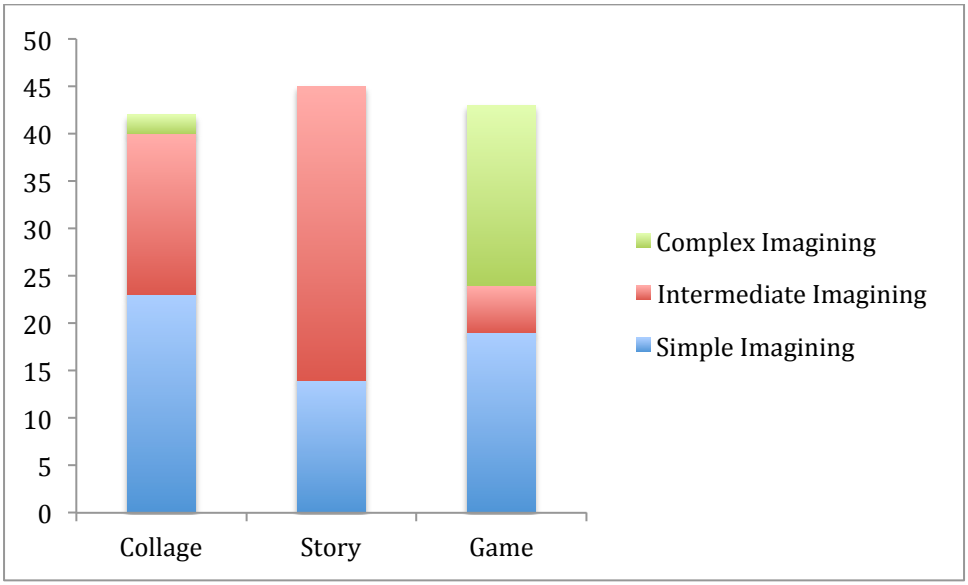
Figure 16 – Frequencies of Levels of Demonstrating Projects Depicted in Videos



Each of the Code and Tell interview videos was placed in a category based on what level of demonstrating projects it exemplified during that video. Figure 16 shows the overall patterns of frequencies of each level of demonstrating projects as they relate to the interview videos depicting student presenters talking about specific animated genre projects. A look at the chart shows that students demonstrated projects in a simple manner the most during collage project interviews and this decreased throughout the rest of the curriculum. The chart also shows that intermediate demonstrating occurred most frequently during the story project interviews. Finally, complex demonstrating occurred the most during game project interviews.

Patterns in Levels of Imagining Projects

Figure 17 – Frequencies of Levels of Imagining Projects Depicted in Videos



Each of the Code and Tell interview videos was placed in a category based on what level of imagining projects it exemplified during that video. Figure 17 shows the overall patterns of frequencies of each level of imagining projects as they relate to the interview videos depicting student presenters talking about specific animated genre projects. As depicted in the chart, roughly half of the participants imagined projects in a simple manner throughout the curriculum. Imagining in an intermediate manner was most frequent during the collage project interviews and then declined. Imagining in a complex manner increased from barely seen to between half and a third of the participants after the collage project interviews.

Discussion of Patterns in Whole Group of Students

Videos from the Code and Tell interview were categorized based on level of describing, demonstrating, and imagining projects in order to better understand the mediating process of using computational thinking. This mediating process could potentially lead to the desired outcome of the learning design, development of students' computational thinking abilities, which

would address this thesis's overarching research question, how does the Code and Tell activity provide opportunities to learn computational thinking in the early childhood classroom?

The patterns in levels of describing projects are unsurprising. Collage projects only require putting characters on a screen, story projects require characters to carry out plot points of some kind, and game projects require characters to have some sort of function. For this reason one would expect simple describing, which is exemplified by students explaining what project elements are, as being most commonly displayed during collage project interviews. The characteristics of the story project could explain why intermediate describing, pointed to by students explaining what their project elements do, is the most commonly displayed during the story project interviews. Finally, the traits of the game project may explain why complex describing, defined by students explaining the purpose or use of project elements, is most commonly seen in the game projects.

The patterns in levels of demonstrating projects are unsurprising as well. Students showing individual project elements separately defines simple demonstrating. Since the collage project is introduced as an activity where students combine different things on a page, without necessarily drawing from a theme, this level of demonstrating would make sense as the most commonly seen during collage project interviews. Once students reach the story project, they must tie events together to create a cohesive narrative. Demonstrating their project elements in a way that shows they are related in a system would make sense for demonstrations of story projects. Finally, making a ScratchJr game requires building in some sort of user interaction so it makes sense that more students displayed complex demonstrating during game project interviews, which was defined earlier as showing project elements as an interactive system.

The patterns in levels of imagining projects are more interesting. Throughout the whole curriculum, about half of the participants exhibited imagining during their interviews in a way that reflects the simple level. This means that as they talked about possibilities and next steps, they focused on physical project elements rather than conceptual ones or how the audience or user perceived the project. About half of the collage project interviews showed students imagined in an intermediate way, meaning they focused on conceptual elements of their projects. For the story projects and game project, both interview sets saw students imagining in a complex manner, focusing on audience or users.

Categories of Students

In addition to looking at the distribution of different levels of describing, demonstrating, and imagining in the samples of collage, story, and game project interview videos for the selection of students as a whole, individual student categorizations by level over the course of their three interviews were explored. A holistic qualitative analysis combined with field notes taken during the curriculum enactment by ScratchJr tutors brought to bear that the categories used earlier to describe behavior in interviews could also be used to describe the way some students tend to most naturally communicate about their projects. The following case studies present illustrations of the child who describes, the child who demonstrates, and the child who imagines. The quotations and scenes draw from the interview videos as well as field notes. These three categories do not necessarily apply to all of the students who participated in the Code and Tell study but each of these categories represents a plurality of students within the sample. Following the case studies, this conceptual framework as a whole is discussed.

The Child Who Describes

Vignette – Denise

When Denise's collage project interviewer hits record, she is ready to talk about anything and everything involving a collage project with a big family of characters and their home. She begins by talking about how she acquired all of her characters. "I went to, like, the plus sign, which was in the box. I got all the people I wanted. For example, mother, baby…" She trails off, seemingly losing her train of thought, but then after poking a couple of her characters on the screen, starts back up again talking about the process of coloring characters with the ScratchJr paint editor. "For example, if you go here, you can choose different colors. So I chose all these colors and pressed check." Transitioning to the next phase of her design process, positioning her different characters, she continues, "Then I wanted to put my baby, I think I should put it here." She moves the baby about half an inch upwards. "Then I got this house." For a few minutes, Denise explains all the intricate details, and eventually the interviewer cuts her off.

Some time later, Denise is back to interviewing. This time, she presents a story project. "This is a project about a cat. So, in the beginning, he was telling all about his owner." She taps the green flag to show a cat and owner speaking to each other next to a house in a field. The cat says, "Hi. I am Mr. Cat. I like to jump around and I have an owner," and the owner says, "Hi. I'm [name redacted]." Denise beams and adds, "I didn't name it yet but I know what I'm going to name it. I'm going to name it, 'The Adventures of [name redacted] and Mr. Cat.'" Her interviewer asks her how she made her project and Denise launches into the details, "I just, like, used all these arrows and all these movements, and also a lot of type of things..." She pauses and turns her attention to her main character. "The cat was supposed to say mostly everything

because he wanted to tell about his owner." At this point, nearly five minutes have gone by and the interviewer decides to wrap up.

Denise returns once more to the interviewing activity with a game project. She shows the camera her guessing game, where the player has to tap characters on the screen decorated like fairies until he or she discovers the one that dances. She explains, "So the way you play this is you gotta find the dancing teen. You need to press one, so let's say I press this one." After quickly demonstrating the gist of the game, she shuffles the characters around. "Try to guess which one it is. I don't know which one it is. And I got it so this one is the winner." She points out, "Only this one can fly," noting that she is trying to trick the player into tapping the character with wings because it stands out, when in reality a different character is the one that dances.

The child who describes is ready to talk about anything and everything when it comes to their ScratchJr project. Although he or she realizes the importance of showing how a project works, he or she relies on words to set up an audience's perception of the project in a more personally meaningful way. Early on during the Code and Tell interviews, the child who describes may be seen holding up his or her iPad and talking all about how he or she made different elements of their project look a certain way or how ideas led to each other. After the polar bear came the fish, and after the fish came the bird, for example. As this student becomes more accustomed to presenting ScratchJr projects in interviews and develops his or her computational thinking abilities, he or she spends more time explaining how things work or how to use the project effectively. The demonstrations may occur more often but communication about a project's essence tends to lead with verbal illustrations.

The Child Who Demonstrates

Vignette – Jason

During Jason's collage interview, his partner asks him to start telling him about his project. He hastily responds, "Well, it's really crazy…" clearly fixated on making a few last changes to his project rather than engaging with the interview right away. His partner responds, "Please can you look at me so I can see you?" Jason stares at his iPad and his fingers fly. He gives another cursory preview with his words, claiming, "I made two bars of weird settings," referring to two very long scripts he wrote for one of his characters. Finally, after much deliberation, he reveals the project to the camera. In it, we see one of the busiest projects out of the entire class. A car flies across a space background while rotating, and resizing. Behind it, an astronaut, a rocket ship, a rainbow cat, and some other miscellaneous additions populate the screen. Jason does not have much to say but three seconds viewing the action on the screen says it all.

Later on, during Jason's story interview, he engages the interviewer much sooner than he did during his collage interview. But in the typical fashion of the child who demonstrates, he gets distracted halfway through his descriptions by the action on the screen and ensuring that what he has programmed is shown adequately. The interviewer says, "Tell me about your project," and Jason responds, "Well, some parts that are kind of weird, like the part where the tuna fish bowl grows so big…" He pauses his speech to show the entirety of an animation where a customized tuna fish bowl, indeed, grows to its maximum size. In the background the text reads, "Pat eats his food," referring to the cat character in the middle of the screen. About a dozen personally colored black and white cats on the left side of the screen ask if they can eat some as Pat comments on the delicious tuna fish. As these things happen Jason homes in on the exciting

focal point of the massive growth. He then skips over a page of his story without action and exclaims, "My favorite part is when the cat really does sword fighting with the doggy." He pauses, turning the screen to face the camera perfectly to show a new page where a cat and dog with swords battle all over the stage to a symphony of cartoon pop sounds. During each of the following questions, Jason takes more opportunities to reshow his demonstrations.

For the game project, Jason created a game where a cat character can, in a series of pages that represent levels, be controlled to battle a custom made human character, avoid bombs, and take part in two other somewhat unclear adventures. His answer to the first interview question is more specific this time, "You're trying to not get hit," referring to the challenge of the first level. He demonstrates each section of his game for several seconds and the interviewer transitions into the last two questions. "What would you do if you had more time?" the interviewer asks. He quickly shares his answer, to put more bombs on the page and make the game more difficult, and then gets back to the demonstrations. As he taps through his last demonstration, he mutters about things that do not work and need to be fixed. "It's laggy. It's pretty slow." For the third time, it is clear that Jason could iterate on his design and create wowing demonstrations for a long time but the interviewer calls it a day.

The child who demonstrates works meticulously on projects to completion and prefers to show things happening on the screen rather than talking about them. He or she may be comfortable talking about how a project works or what concepts drove its creation but the project itself, in all its animistic glory, usually takes precedent if it can be physically presented. Rather than talking about twenty pigs falling from the sky, he or she will show you what it looks like and giggle alongside you. As the child who demonstrates begins participating in the Code and Tell activity, he or she describes projects as an afterthought or answers questions to keep his or

her interviewer at bay while making sure the demonstrations can operate smoothly. Later, as the child grows more adept at computational thinking, he or she develops a way to speak in tandem with the demonstrations, offering little bits of insight as the screen captures the interviewer's attention and taking on the role of the user or audience to present an authentic experience of the project. When the child who demonstrates needs to communicate the essence of their project, they focus on action and execution.

The Child Who Imagines

Vignette – Lisa

Lisa opens her collage interview excitedly talking about the scene she programmed with a few custom colored cat characters and a soccer ball in a forest. She reports, words spilling out of her mouth rapidly "So, my project takes place in the woods. There are four cats—a mom, a dad, a baby and a kid. They're playing soccer and they're trying to get the kid into the soccer game." In response to the interviewer asking her what she would do with more time, she replies before he even finishes asking, "I would add more cats to make them into soccer teams." Although there appear to be few programming blocks, Lisa asserts what is happening with her words.

Back for more interviewing with her story project, Lisa explains, "My project is about a cat saying, 'Snow,' and on the next page on my picture there's, like, a cat and he's, like, predicting the future, and then the next scene he's asking if somebody will play with him, and then it's basically the end." She gusts through her demonstration, allowing the audience to see that by "predicting the future" she means the background has changed from autumn to winter between two pages. Two cats appear on the screen in scarves of different colors and they engage in dialogue that suggests they are about to play together in the snow. The interviewer pauses for

a moment and inquires about future design decisions. Lisa adds, "Um, I think I would have added a lot more stuff to the page where he's asking if somebody would play with him."

In her game project, Lisa presents three different games, each on their own page. "So, um, I had different levels in my project. In this level, you have to find six animals and put them inside this thing." She points to a blue spiral at the corner of her screen and drags several miniature animals into it, watching them disappear one by one. She turns the page and continues, "And then this project, um, there are things that keep on going up and down. You try to get your cat to the end and every time he hits the fire he'll have to go back to the beginning where he started from." She moves the cat to its goal destination and the page turns again. "In this level, there are these cats that keep on getting bigger and smaller. Your job is to get your cat to get to one of these things. If you bump anything, you start from the beginning." Each individual game appears to be incomplete but the ideas are all perfectly intact as Lisa describes them.

The child who imagines bubbles over with ideas and concepts, not necessarily caring whether they are executed on the screen or not. The important thing is that the ideas are conveyed to the audience and some semblances of them are portrayed in ScratchJr. In early ScratchJr interviews, this kind of student fills in blank programs with possibilities for action, dialogue, and relationships between characters. He or she offers future directions or imposes concepts to guide the physical happenings on the screen. Later, as his or her computational thinking abilities grow, interviews portray the child who imagines dreaming up possibilities and programming just enough to make them believable. The child who imagines conveys the essence of their ScratchJr project by combining what is seen with what *could* be seen.

Discussion of Categories of Students

The child who describes, the child who demonstrates, and the child who imagines represent three pluralities of students in the early childhood classroom using ScratchJr and presenting projects in Code and Tell interviews. While each student can grow to be adept in many different styles of conveying their ScratchJr projects to an audience, the child who describes tends to favor verbalizing, the child who demonstrates tends to favor executing, and the child who imagines tends to favor theorizing. These illustrations, drawn from a handful of interviews and field notes, do not account for all students nor do they necessarily represent the absolute image of a child in each category. What they do show, however, is a distinct pluralism in the trajectories of young children learning about computational thinking and how signs of development of computational thinking abilities can manifest themselves in a myriad of ways. Perhaps more importantly, these manifestations are all somewhat identifiable from artifact-based peer video interviews, as exemplified by the vignettes provided above. A high-level takeaway from this finding is that early childhood educators can watch for a variety of different learning paths in their classrooms that all have ties to the development of computational thinking. They can support and encourage growth along these different lines as well as help students understand that these differences stem from personal styles rather than deviations from proper learning of computational thinking.

The purpose of this thesis was to explore how the Code and Tell activity could support opportunities for students to learn computational thinking in the early childhood classroom. Specifically, in terms of Sandoval's (2014) conjecture map, using computational thinking during the Code and Tell activity was hypothesized as a mediating process for the development of computational thinking and this study was meant to elicit the nature of this potential process. The

Code and Tell activity helps make the pluralism of learning computational thinking visible and may provide educators with a way to begin seeing and understanding the nature of children who describe, demonstrate, and imagine during interviews (and perhaps beyond them). In doing so, it may mediate the development of computational thinking with ScratchJr in the early childhood classroom.

6. Limitations

School Setting

There were several limitations to this study due to the nature of conducting research in a school setting. First of all, the participants did not all receive the same treatment during the study. Several students were absent on at least one of the days during which the curriculum was taught. While the ScratchJr tutors did their best to make sure these students received enough individual attention in order to catch them up when they returned to class, this personalized teaching cannot be equivalent to experiencing the curriculum alongside peers as originally planned. Second, the study was conducted in three different classrooms and the behavior of the regular classroom teachers varied between these settings. Since regular classroom teachers were encouraged to help facilitate classroom management and take part in teaching ScratchJr at their individual comfort levels, students in different classrooms experienced the curriculum enactment differently with regards to what extent their individual teachers were involved. Finally, due to scheduling limitations and a limited number of ScratchJr tutors, different classes would participate in the same lessons at different times and usually on different days. Intervals between lessons were not always consistent across the three classes and circumstances specific to certain days sometimes affected one class but not another (e.g. one class participated in a lesson on the same day that most of the students were in costume for a Halloween celebration).

First Code and Tell Study

This study marks the first time that the Code and Tell activity was used in an early childhood classroom and there are many ways that students could have been better prepared for the activity to make it as effective as possible. First, there were many videos from the interviews where the camera did not effectively capture what was happening on the presenter's screen. Doing a practice interview activity where students learn about how to film their partners in a way that allows them to successfully record their presenter's projects, descriptions, and demonstrations could drastically improve the quality of the videos. Second, while some students provided rich descriptions and demonstrations for their interview videos, others provided very little. Besides the possibilities that these students were less articulate or interested in the activity, these students may have been confused about what the activity was for. Weaving the videos into a meaningful and repeated classroom activity throughout the curriculum, such as a "viewing day," could help some of the more unforthcoming students find a reason to participate at a higher level of engagement in the Code and Tell activity.

Lost Projects

Occasionally, due to unknown circumstances, students' ScratchJr projects would be lost from their iPads between the day that they created them and the day that they needed to present them in Code and Tell interviews. This was an uncommon occurrence but did happen a few times to students unbeknownst to them until they opened their iPad to the ScratchJr app at the beginning of an interview lesson. When this mishap occurred, in order to keep the method as consistent as possible, these students were instructed to create a new project and to do their interview toward the end of the lesson period so they would have as much time as possible to

create a meaningful ScratchJr project. Despite best efforts, these lost projects may have affected the results.

7. Future Directions

This study was only an exploratory one and there are plentiful directions for further research on the learning opportunities that the Code and Tell activity in the early childhood classroom using ScratchJr.

Dyads

This study did not investigate the effect of partnerships on how students behaved during their interviews or how they learned computational thinking. Certainly, there are many ways that dyads could be explored in order to reach a more comprehensive understanding of the nature of the Code and Tell activity and its potential for learning. The following questions can serve as jumping off points for these types of studies:

- How does the presenter's relationship with the interviewer affect his or her behavior during the Code and Tell activity?
- How would changing partners during each interview affect the outcomes of the activity?
- What if students pick their partners instead of their regular classroom teachers assigning them?
- Do the two partners within a dyad have similar trajectories of learning throughout a curriculum?
- How does an interviewer's behavior of interrupting the presenter affect the presenter's behavior and learning?

Reviewing Videos

Although students had the opportunity to participate in the Code and Tell activity three different times throughout the curriculum enactment, they never had a formal opportunity to review their videos and learn from them nor the opportunity to see their classmates' videos. Understanding the affordances of different ways students could engage with interview videos after the interviews themselves could be useful for classroom practice. The following activities could give students more opportunities to learn from the Code and Tell activity:

- The regular classroom teacher could select a few videos each week to show the class and lead a discussion about how to effectively describe and demonstrate projects for the video audience.
- Students could be given multiple "takes" to film their interview, compare them together, and then select the ones they want to keep.

Interview Questions

There are several ways that the interviews could be changed in order to evoke student learning most effectively in the Code and Tell interviews. In particular, one surprising finding from the study was that students rarely talked about ScratchJr programming blocks during their Code and Tell interviews. If researchers are interested in capturing students understanding of computational thinking in a way that is more situated in the programming language itself, new questions may need to be added to the interview process. The following questions could be tested to see how students talk about the ScratchJr programming language during their Code and Tell interviews:

- What blocks did you use to program your characters?
- Why did you choose those programming blocks?

- Are there other ways to program your characters?
- Did your programs change as you worked on your project?
- How difficult was it for you to program your characters?

Comparing Code and Tell to Other Assessments

Previous ScratchJr research has investigated the use of project portfolios and Solve It programming challenges as artifacts for assessing students' learning of computational thinking. Evaluating the Code and Tell activity as a computational thinking assessment was beyond the scope of this study but there is some promise that it could be used as a way to supplement these other two forms of assessment. Further research could compare the affordances of assessing student learning with these three different modes.

References

Barr, V. & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads, 2*(1), 48-54.

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing Computational Thinking Patterns. Paper presented at the Special Interest Group on Computer Science Education (SIGCSE) Conference, Dallas, TX.

Bers, M. (2006). The role of new technologies to foster positive youth development. *Applied Developmental Science*, *10*(4), 200-219.

Bers, M. (2008). *Blocks to Robots: Learning with technology in the early childhood classroom.* New York: Teachers 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., Ponte, I., Juelich, K., Viera, A., & Schenker, J. (2002) Teachers as Designers: Integrating Robotics into Early Childhood Education. *Information Technology in Childhood Education,* 123-145.

Bers, M.U., Flannery, L.P., Kazakoff, E.R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education, 72,* 145-157.

Brennan, K., & Resnick, M. (2012). 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. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, *2*(2), 141-178.

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.

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

Collins, A. (1992). Towards a design science of education. In E. Scanlon & T. O'Shea (Eds.), New directions in educational technology (pp. 15-22). Berlin: Springer.

Denner, J., Wener, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, *58*(1), 240-249.

Dey, I. (1993). *Qualitative Data Analysis : A User-Friendly Guide for Social Scientists*. London: Routledge.

Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., Kleinsasser, W., Dehlinger, J. & Kaza, S. (2011). A model for piloting pathways for computational thinking in a general education curriculum. In the Proceedings of the 42nd ACM technical symposium on Computer Science education, Dallas, TX, 257-262.

Dwyer, H., Hill, C., Carpenter, S., Harlow, D., Franklin, D. (2014). Identifying elementary students' pre-instructional ability to develop algorithms and step-by-step instructions. In the Proceedings of the 25th ACM technical symposium on Computer Science education,

Atlanta, GA, 511-516.

Fields, D.A., Searle, K.A., Kafai, Y.B., & Min, H.S. (2012). Debuggems to assess student learning in e-textiles. In Proceedings of the 43rd SIGCSE Technical Symposium on Computer Science Education, Raleigh, NC.

Flannery, L.P., Kazakoff, E.R., Bontá, P., Silverman, B., Bers, M.U., & 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.

Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., Dreschler, G., Aldana, G., Almeida-Tanaka, P., Kiefer, B., Laird, C., Lopez, F., Pham, C., Suarez, J., Waite, R. Assessment of computer science learning in a scratch-based outreach program. In Proceedings of the 44th SIGCSE Technical Symposium on Computer Science Education, Denver, CO, 371-376.

Han Koh, K., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Toward the automatic recognition of computational thinking for adaptive visual language learning. In *Proceedings of the 2010 Conference on Visual Languages and Human Centric Computing (VL/HCC 2010).* (pp. 59-66). Madrid, Spain: IEEE Computer.

ISTE & Computer Science Teachers Association (2011). Operational definition of computational thinking for K–12 education.

Jones, S. (1985). The Analysis of Depth Interviews. *Applied Qualitative Research,* London: Gower.

Kazakoff, E., & 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.

Landivar, L. C. (2013). *Disparities in STEM Employment by Sex, Race, and Hispanic Origin*.

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

Lerner, R.M., Dowling, E.M., & Anderson, P.M. (2005). Positive youth development, a developmental systems view. In C.B. Fisher & R.M. Lerner (Eds.). *Encyclopedia of applied developmental science* (pp. 859-862). Thousand Oaks, CA: Sage Publications.

Miles, M.B., Huberman, A.M., & Saldaña, J. (2013). *Qualitative Data Analysis: A Methods Sourcebook*. Thousand Oaks, CA: Sage Publications.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York, Basic Books.

Papert, S. An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Perlis, A. (1962). The computer in the university. In M. Greenberger (Ed.), *Computers and the World of the Future*, (180–219). Cambridge, MA: MIT Press.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*, 265-269. New York, NY: ACM Press.

Rushkoff, Douglas. (2010). *Program or be Programmed: Ten Commands for a Digital Age.* Cambridge, MA: MIT Press.

"About ScratchJr." Retrieved September 18, 2014, from http://www.scratchjr.org/about.html.

Seiter, L., & Foreman, B. Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the 9th annual international ACM conference*

on International computing education research (ICER). ACM, New York, NY, USA, 59-66.

Stolee, K. T., & Fristoe, T. (2011). Expressing computer science concepts through Kodu game lab. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE)*. ACM, New York, NY, USA, 99-104.

Strawhacker, A. L., & Bers, M. U. (2014). ScratchJr: Computer Programming in Early Childhood Education as a Pathway to Academic Readiness and Success. *Poster presented at DR K-12 PI Meeting*, 5 August 2014, Washington D.C.

Piaget, J. (1929). *The child's conception of the world*. London, Routledge & Kegan Paul.

Portelance, D. J., Strawhacker, A. L., Bers, M. U. (2014) Constructing the ScratchJr programming language in the early childhood classroom. Manuscript in preparation.

Tudge, Jonathan. (1992). Vygotsky, the zone of proximal development and peer collaboration: Implications for classroom practice. In L. C. Moll (Ed.), *Vygotsky and Education: Instructional Implications and Applications of Sociohistorical Psychology.* Cambridge, UK: Cambridge University Press.

Turkle, S. & Papert, S. (1990). Epistemological Pluralism and the Revaluation of the Concrete. *SIGNS: Journal of Women in Culture and Society, 16*(1), 128-157.

Vygotsky, L. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The Fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*, 215-220. New York, NY: ACM.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age, Association for Computing Machinery and Computer Science Teachers Association.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *Journal of the Learning Sciences*, *17*(4), 517-550.