Computational Thinking Unplugged: Comparing the Impact on Confidence and Competence from Analog and Digital Resources in Computer Science Professional Development for Elementary Teachers

Christopher Harris
St. John Fisher College, chris@playplaylearn.com

Follow this and additional works at: https://fisherpub.sjfc.edu/education_etd

Part of the Education Commons

Recommended Citation

Please note that the Recommended Citation provides general citation information and may not be appropriate for your discipline. To receive help in creating a citation based on your discipline, please visit http://libguides.sjfc.edu/citations.

This document is posted at https://fisherpub.sjfc.edu/education_etd/374 and is brought to you for free and open access by Fisher Digital Publications at St. John Fisher College. For more information, please contact fisherpub@sjfc.edu.
Computational Thinking Unplugged: Comparing the Impact on Confidence and Competence from Analog and Digital Resources in Computer Science Professional Development for Elementary Teachers

Abstract
The demand for computer science instruction is increasing across the K-12 spectrum, but in many cases elementary teachers are ill prepared to teach the subject. Based on prior research showing a preference for analog interfaces, this study compared the impact of analog and digital interface modalities on teachers’ confidence and competence gains in professional development on computational thinking conceived within the framework of cognitive acceleration. The analog group used the Robot Turtles board game and the digital group used the Scratch Jr. app on iPads while receiving the same professional development content. A single-case experimental design approach with a multiple-baseline approach to establish control and appropriate randomization techniques was used to allow for generalization of findings and identification of a functional relationship. Teachers were assessed using the Elementary Teacher Computer Programming Self-Efficacy Scale for confidence and the Computational Thinking Test for competence. The results indicated a significant and higher effect size on confidence for the analog cases as compared to the digital. Visual analysis confirmed these findings and provided emerging support for a functional relationship. Recommendations for modifications to current professional development, classroom instruction, and policy making practices to adopt an analog-first approach to computer science based on the foundational concepts of computational thinking were identified based on these findings.

Document Type
Dissertation

Degree Name
Doctor of Education (EdD)

Department
Education

First Supervisor
Guillermo Montes

Second Supervisor
Bernard Ricca

Subject Categories
Education

This dissertation is available at Fisher Digital Publications: https://fisherpub.sjfc.edu/education_etd/374
Computational Thinking Unplugged: Comparing the Impact on Confidence and Competence from Analog and Digital Resources in Computer Science Professional Development for Elementary Teachers

By

Christopher Harris

Submitted in partial fulfillment of the requirements for the degree Ed.D. in Executive Leadership

Supervised by
Dr. Guillermo Montes

Committee Member
Dr. Bernard Ricca

Ralph C. Wilson, Jr. School of Education
St. John Fisher College

August 2018
Biographical Sketch

Christopher Harris currently serves as the Director of the School Library System for the Genesee Valley Educational Partnership, an educational services agency in Le Roy, NY. Mr. Harris attended Hobart and William Smith Colleges from 1994 to 1998, graduating *cum laude* in 1998 with a Bachelor of Arts in American Studies. He attended North Carolina State University from 2000 to 2001 and graduated with a Masters of Education with a focus in instructional technology in 2001. Mr. Harris attended the State University of New York College at Brockport from 2005 to 2006 and graduated with a Certificate of Advanced Study in Educational Administration. Concurrently, he attended the State University of New York University at Buffalo from 2005 to 2008 and graduated in 2008 with a Masters of Library Science. Mr. Harris began doctoral studies at St. John Fisher College in the summer of 2016 in the Ed.D. Program in Executive Leadership. He pursued his research on increasing teacher confidence with computational thinking under the direction of Dr. Guillermo Montes and Dr. Bernard Ricca and received the Ed.D. degree in 2018.
Abstract
The demand for computer science instruction is increasing across the K-12 spectrum, but in many cases elementary teachers are ill prepared to teach the subject. Based on prior research showing a preference for analog interfaces, this study compared the impact of analog and digital interface modalities on teachers’ confidence and competence gains in professional development on computational thinking conceived within the framework of cognitive acceleration. The analog group used the Robot Turtles board game and the digital group used the Scratch Jr. app on iPads while receiving the same professional development content. A single-case experimental design approach with a multiple-baseline approach to establish control and appropriate randomization techniques was used to allow for generalization of findings and identification of a functional relationship. Teachers were assessed using the Elementary Teacher Computer Programming Self-Efficacy Scale for confidence and the Computational Thinking Test for competence. The results indicated a significant and higher effect size on confidence for the analog cases as compared to the digital. Visual analysis confirmed these findings and provided emerging support for a functional relationship. Recommendations for modifications to current professional development, classroom instruction, and policy making practices to adopt an analog-first approach to computer science based on the foundational concepts of computational thinking were identified based on these findings.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biographical Sketch</td>
<td>........................................................................................................... iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>.................................................................................................................... iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>.................................................................................................................. v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>.................................................................................................................. viii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>................................................................................................................... ix</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>........................................................................................................... 1</td>
</tr>
<tr>
<td>Problem Statement</td>
<td>........................................................................................................... 2</td>
</tr>
<tr>
<td>Theoretical Rationale</td>
<td>........................................................................................................... 9</td>
</tr>
<tr>
<td>Statement of Purpose</td>
<td>........................................................................................................... 13</td>
</tr>
<tr>
<td>Research Questions</td>
<td>........................................................................................................... 14</td>
</tr>
<tr>
<td>Potential Significance of the Study</td>
<td>........................................................................................................... 14</td>
</tr>
<tr>
<td>Definitions of Terms</td>
<td>........................................................................................................... 15</td>
</tr>
<tr>
<td>Chapter Summary</td>
<td>........................................................................................................... 16</td>
</tr>
<tr>
<td>Chapter 2: Review of the Literature</td>
<td>........................................................................................................... 17</td>
</tr>
<tr>
<td>Introduction and Purpose</td>
<td>........................................................................................................... 17</td>
</tr>
<tr>
<td>Computational Thinking in K-12</td>
<td>........................................................................................................... 19</td>
</tr>
<tr>
<td>Competence and Confidence in Computational Thinking</td>
<td>........................................................................................................... 22</td>
</tr>
<tr>
<td>A Play-Based Approach to Computational Thinking</td>
<td>........................................................................................................... 26</td>
</tr>
<tr>
<td>Interface Modalities and Playful Approaches to Computational Thinking</td>
<td>........................................................................................................... 30</td>
</tr>
<tr>
<td>Board Games as Analog, Playful Computational Thinking</td>
<td>........................................................................................................... 37</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Gaps and Further Research</td>
<td>44</td>
</tr>
<tr>
<td>Chapter Summary</td>
<td>48</td>
</tr>
<tr>
<td>Chapter 3: Research Design Methodology</td>
<td>49</td>
</tr>
<tr>
<td>Introduction</td>
<td>49</td>
</tr>
<tr>
<td>Research Design</td>
<td>50</td>
</tr>
<tr>
<td>Research Context</td>
<td>52</td>
</tr>
<tr>
<td>Research Participants</td>
<td>53</td>
</tr>
<tr>
<td>Instruments Used in Data Collection</td>
<td>54</td>
</tr>
<tr>
<td>Procedures</td>
<td>59</td>
</tr>
<tr>
<td>Analysis</td>
<td>61</td>
</tr>
<tr>
<td>Chapter Summary</td>
<td>63</td>
</tr>
<tr>
<td>Chapter 4: Results</td>
<td>64</td>
</tr>
<tr>
<td>Research Questions</td>
<td>64</td>
</tr>
<tr>
<td>Visual Analysis</td>
<td>65</td>
</tr>
<tr>
<td>Statistical Analysis</td>
<td>77</td>
</tr>
<tr>
<td>Summary of Analysis</td>
<td>79</td>
</tr>
<tr>
<td>Chapter 5: Discussion</td>
<td>81</td>
</tr>
<tr>
<td>Introduction</td>
<td>81</td>
</tr>
<tr>
<td>Implications of Findings</td>
<td>83</td>
</tr>
<tr>
<td>Recommendations and Future Research</td>
<td>93</td>
</tr>
<tr>
<td>Limitations</td>
<td>101</td>
</tr>
<tr>
<td>Conclusion</td>
<td>102</td>
</tr>
<tr>
<td>References</td>
<td>104</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Item</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.1</td>
<td>Influences in Cognitive Acceleration</td>
<td>10</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Mean Competence Results</td>
<td>67</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Within Condition Statistics for Jumper</td>
<td>70</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Within Condition Statistics for jlo</td>
<td>72</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Within Condition Statistics for Taylor</td>
<td>74</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Within Condition Statistics for Charlotte</td>
<td>76</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>Between Condition Statistics for all Cases</td>
<td>77</td>
</tr>
<tr>
<td>Table 4.7</td>
<td>Two-Sample T-Test and Effect Size Calculations</td>
<td>79</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Item</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>Visitor Interaction by Interface</td>
<td>35</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Directions from <em>Robot Turtles</em></td>
<td>42</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Mean Competence Scores</td>
<td>68</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Jumper: Confidence and Competence</td>
<td>70</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>jlo: Confidence and Competence</td>
<td>72</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Taylor: Confidence and Competence</td>
<td>74</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Charlotte: Confidence and Competence</td>
<td>75</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Analog Group Comparison</td>
<td>85</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

When computers began to enter general use in the 1950s, to be a computer user was to be a computer scientist. In the age of punch cards and other physical input systems, operations and computations were directly programmed into the computer by the user. Over time, developments in software and hardware added new layers of abstraction between the user and the machine. These technical advancements, along with other factors, created a separation between the roles of computer user and computer scientist and led to a decline in K-12 computer science instruction. Accompanying that was a similar decline in professional development and pre-service instruction for teachers around computer science (Niess, 1990). Recently, computer science has enjoyed a return to prominence. Efforts to increase computer science instruction for students through programs like Hour of Code have been successful in providing students a first experience with code. In contrast, little attention has given to preparing in-service teachers to lead ongoing instruction around computer science (Menekse, 2015). This issue is most critical at the elementary level where a lack of teacher confidence and competence regarding computer science could harm student attitudes towards the subject later (Duncan, Bell, & Tanimoto, 2014). The theory of cognitive acceleration – a synthesis of the cognitive development theories of Piaget and Vygotsky that proposes a way to support more effective learning – will be applied as a framework for understanding how teacher development might be implemented concurrently with student instruction.
Problem Statement

Computer science instruction is increasing in the United States because of mounting pressure to fill high-tech jobs as well as increased instructional support resources such as Code.Org, but in-service elementary teachers often do not have the requisite confidence and competence to be effective leaders in this subject (Gallup & Google, 2016). While historical factors have led to a lack of computer science instruction and professional development across the K-12 spectrum, the problem is especially significant in elementary schools where formal instruction and informal computer science opportunities are much less frequent than in high schools (Gallup & Google, 2016). In a survey by Gallup and Google (2016), only 40% of elementary schools offered any type of computer science instruction compared to 78% of high schools. Elementary schools were also much less likely (44%) than high schools (63%) to have a computer science club or activity option (Gallup & Google, 2016). A lack of qualified teachers was reported by Gallup and Google (2016) as the primary reason for not offering computer science instruction. There is some research on the efficacy of approaches for training pre-service teachers on computer science (Cetin, 2016), but the needs of in-service teachers who will be expected to meet the rising demand for instruction are not addressed in the literature.

There is an almost total gap in the literature around computer science professional development for in-service elementary teachers (Menekse, 2015). The problem of teacher preparation around computer science seems to be most pronounced at the elementary level. In a review of 82 identified research studies from the United States addressing professional development for teachers on computer science between 2004 and 2014, Menekse (2015) found no studies on professional development created solely for
elementary teachers, and only three studies on professional development programs that included elementary teachers along with all other grade levels. A preliminary report from a pilot project by Duncan, Bell, and Atlas (2017) in New Zealand explained the cause for concern: “The teachers we have worked with throughout this, and previous studies, were more often than not anxious about teaching CS and programming concepts, and think they are not capable of doing this well” (p. 5). However, Duncan et al. (2017) did note that the teachers were capable of teaching computer science given appropriate levels of support. The nature of the support and the potential effect of different types of supports are not addressed. Determining the types of support that are most effective for teachers is therefore a critical issue.

**Historical development of problem.** The history of computer science in K-12 schools can help explain the lack of attention given the subject in recent years. Beyond the changes in software and hardware design that led to a separation of users and programmers (Birnbaum, 1982), two other factors have influenced the path of computer science adoption in schools. First is the shifted definition of computer literacy influenced by an information and communications technology (ICT) use approach (Niess, 1990). Second, there has been confusion around the placement of computer science within the established academic environment. These factors were explored within the context of the United States and international computer science policy and curriculum (Adrion, Fall, Ericson, & Guzdial, 2016).

**Users and programmers.** On a technical level, as hardware became increasingly miniaturized and integrated it was harder for users to program and interact directly with the computer on a physical level. Software, a set of pre-written instructions to deliver
programming to the processor and other components, replaced hardware as a primary selling point for computers. A survey of computer trends by Birnbaum (1982) documents this shift both in terms of the rising costs of software as the largest percentage of technology expenditures and in early recognition of the potential for software like VisiCalc, the first modern spreadsheet. This new type of software and the accompanying development environment, Birnbaum (1982) explained, “frees the application developer or end user from concern with the details of program construction” (p. 763). Unlike earlier computer users who had to also be computer scientists and programmers, new software-based computer users could simply use programs written by others in a more task-oriented environment. These technical developments were changing the landscape just as K-12 schools were first working to bring computers to the classroom and likely influenced the shift away from programming and computer science.

**Redefining computer literacy.** Initial efforts to introduce computers into K-12 classrooms incorporated the dual roles of user and programmer. Seymour Papert’s (1980) use of LOGO, a programming language designed for children, in elementary classrooms was built from a foundation of computer science instruction. Influenced by Piaget’s work on early child development, Papert’s (1980) LOGO language was designed to provide an organic and obvious communication interface between the user and the computer. The goal was computer literacy for both the student and the teacher. This concept was initially defined by one of the co-developers of LOGO, Daniel Watt (1980), as being the ability to program as well as use computers mirroring both the writing and reading aspects of print literacy. Over time, the computer programming aspect of computer literacy was phased
out. In a review of necessary computer competencies for teachers, Niess (1990) dismissed programming skills as outdated and unnecessary for teachers.

By 2000, computer programming was not even included on the survey instrument in a study seeking to rank computer competencies for teachers in Kentucky (Scheffler & Logan, 2000). The competencies instead focused on usage skills like keyboarding, using software, and communicating with email (Scheffler & Logan, 2000). Policy around computer instruction had shifted and as Scheffler and Logan (2000) explained, the intention of the International Society for Technology in Education’s (ISTE) 1998 standards was to “focus on student knowledge and student use of technology rather than what the teacher needs to know about technology and to be able to do with technology” (p. 310). Following the new definition of computer literacy, professional development for teachers also shifted towards basic computer use instead of computer science and programming.

The academic home for computer science. Additional challenges to the introduction of computer science into schools came from within the educational system itself. For example, Forsythe (1967) related the struggle to create a distinct department of computer science at Stanford University that respected the influences of many different existing fields including math, philosophy, and engineering. High schools also struggled to place computers into existing discipline-based departments. Reflecting on the formation of computer science as a discipline, Atchison (1971) also noted ongoing confusion about placing the new department in high schools explaining that the “widest use is for problem solving in mathematics, business and science courses” (p. 132). This explains why many high schools have historically included, and continue to place,
computer science as part of business or career and technical education (Adrion et al., 2016). Without a specific and distinct academic home, computer science has struggled to break into many state standards and curricula and is not prominently tested or included as a graduation requirement (Wilson, Sudol, Stephenson, & Stehlik, 2010). Questions remain in the United States about teacher credentials for computer science courses as well as the placement of computer science as a core classroom subject or a separate special area subject within elementary schools (Adrion et al., 2016; Gallup & Google, 2016). In New York, for example, there is no teacher certification or tenure area for a teacher of computer science.

**Trends in computer science education.** A prominent trend in computer science instruction is the shift towards making computational thinking – the foundational skills underlying computer science and transferable to other disciplines – the instructional core (Voogt, Fisser, Good, Mishra, & Yadav, 2015). Popular and trade publications in recent years have included opinion pieces debating the question of whether every student needs to learn computer science. Megan Smith, former Chief Technology Officer of the United States, called for elementary schools to teach coding stating “second graders learn to read, that’s a perfect time to make them code” (Meyer, 2014, para. 7). Refuting the need for every child to learn coding, Christian Hernandez (2014), a parent and computer science professional from the United Kingdom, instead suggested a broader approach. “Coding refers to the use of a specific computing language to string together instructions for a computing device to execute. Instead, let’s talk about programming: the process and concepts of logic which – when implemented via code – bring digital services to life” (Hernandez, 2014, para. 6). For Hernandez (2014), the critical skill isn’t in writing the
code, but learning computational thinking as the underlying aspects of programming that drive development and implementation of code.

Computational thinking defined. The concept of computational thinking extends back to Papert’s (1980) first use of the term to describe a way of thinking deeply about the abilities of a computer to work and solve problems. More recently, the term was adopted by Wing (2006) in her seminal article defining a different approach to the field of computer science that sought to identify “a universally applicable attitude and skill set” that everyone should learn (p. 33). Modern definitions of computational thinking focus on four concepts: (a) decomposition, or breaking down a problem into parts; (b) pattern recognition, or the ability to interpret data; (c) abstraction, or an understanding of generalized principles; and (d) algorithm design, or the creation of explicit directions for work (Google, n.d.).

Computational thinking is a lens for understanding and viewing the foundational aspects of computer science separated from the application of computer science in writing code. This was described by media theorist Douglas Rushkoff (2011) as invalidating the metaphor about learning programming as being like having to be a mechanic to drive a car. The real comparison, Rushkoff (2011) argued, was that a lack of computational thinking relegated a person to being a passenger in the car instead of the driver. For Rushkoff (2011) computational thinking “is the only way to truly know what is going on in a digital environment, and to make willful choices about the roles we play” (p. 8).

As an approach for computer science instruction, Fletcher and Lu (2009) opined that computational thinking should be taught before students are ever introduced to
computer programming. Starting computer science instruction with programming, Fletcher and Lu (2009) claim, is “akin to teaching basic arithmetic alongside proof construction, and elementary reading and writing with linguistics and discourse analysis” (p. 24). Instead, Fletcher and Lu (2009) argue, an early introduction to the foundational aspects of computational thinking will better prepare students for later success in computer science and other fields.

**International developments.** In the past decade, elementary computer science programs using a computational thinking approach have been successfully implemented on an international level. In 2013-2014, the United Kingdom deployed a national computer science curriculum and mandated instruction for students at all levels. This announcement received a great deal of international attention as it signaled a policy shift from ICT-based computer literacy back to computer science (Fluck et al., 2016). New Zealand added computer science to the national high school curriculum in 2011, and in 2014 revisions to the general curriculum included a focus on computational thinking at all grade levels (Duncan & Bell, 2015). Though not as closely reviewed here due to language issues, computer science is also included as a mandated subject in national curricula for Estonia and Cyprus (Duncan & Bell, 2015) and many other countries are currently working on creating or revising curriculum documents (Fluck et al., 2016). Internationally, then, it can be said that the educational field is moving back to the more holistic approach endorsed by Papert (1980) and other early computer educators who called for teaching the dual roles of computer user and computer scientist.

**Domestic policy and adoption of computer science instruction.** Looking forward in the United States, there is a growing push for computer science instruction.
President Barack Obama launched the Computer Science for All (#CS4ALL) challenge in 2016 calling on schools to implement computer science instruction K-12. In 2017, President Donald Trump extended the call for increased STEM and computer science instruction. External initiatives such as Code.org and the Hour of Code have realized significant penetration into K-12 classrooms since their launch in 2013 with over 870,000 teachers registered to teach the introductory courses (Code.org, n.d.). Additional domestic policy work has been undertaken by the Computer Science Teachers Association that was formed in 2004 as an extension of the Association for Computing Machinery to provide support and advocacy for teachers of computer science. In late 2016, a new K-12 Computer Science Framework was released by a collaborative headed by the Association for Computing Machinery and the Computer Science Teachers Association. The framework provides a roadmap for possible adoption in states or local districts that includes a strong focus on computational thinking. Some states have adopted K-12 computer science curricula or standards: Indiana (April 2016), Massachusetts (June 2016), and Washington (December 2016). South Carolina, Texas (9-12 only), and Florida are currently reviewing draft standards with more states expected to follow. Given the rising demand for computer science instruction and changing curricula, there must also then be a parallel increase in teacher professional development around computer science.

**Theoretical Rationale**

Cognitive acceleration is a theoretical framework for learning within a constructivist environment heavily influenced by, and synthesized from, the work of Piaget and Vygotsky (Shayer, 2005) as seen in Table 1.1. From Piaget, cognitive acceleration has embraced the idea of stages of cognitive development and the need to
base new learning on concrete, often tangible, understandings (Goulding, 2002). For Piaget (1969/2008), learning was broadly defined as the assimilation of new ideas into existing schemata or structures of understanding. This would happen in different ways based on the stage of development in which a child fell as determined mostly by age (Piaget, 1969/2008). In terms of cognitive acceleration, the goal is to facilitate movement from the concrete operational stage into formal operations. The concrete operational stage is thusly named for being based on interactions with tangible objects as opposed to verbal expressions or hypothetical ideations of concepts (Piaget, 1969/2008). A concrete foundation is critical, Piaget (1969/2008) argues, because it allows a child to become “capable of reasoning correctly about propositions he does not believe, or at least not yet” (loc. 1131). The ability to think critically about hypotheticals is a key feature of formal operations. For Piaget (1969/2008) most of this development happens independently for children based on personal experiences.

Table 1.1

<table>
<thead>
<tr>
<th>Cognitive Acceleration Concept</th>
<th>Piaget</th>
<th>Vygotsky</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schemata of formal operations</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Concrete preparation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cognitive conflict</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Metacognition</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Bridging</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Cognitive acceleration adds to Piaget’s work by incorporating Vygotsky’s more socially driven understandings of learning and development. Language was the basis of learning for Vygotsky (1962/2012) in that gaining new vocabulary allowed a child to refine groupings of objects called complexes reminiscent of Piaget’s schemata. Social interactions were the genesis of complexes for Vygotsky (1962/2012): “The child receives all the elements of his complexes in a ready-made form, from the speech of others” (p. 131). The importance of speech and interactions is also seen in Vygotsky’s (1962/2012) idea of the zone of proximal development. There exists, Vygotsky (1962/2012) posits, a zone of ideal learning that engages a child in concepts that are too difficult for him or her to understand in isolation but that can be comprehended based on interactions with an adult teacher. The concept of the zone of proximal development seen in cognitive acceleration in teacher guided challenges that push students to move outside of their comfort zone and in the use of social constructivism to increase understanding at a faster pace through dialogue (Goulding, 2002). Cognitive acceleration also synthesizes Piaget’s emphasis on maturation and development as a source of intellectual development with Vygotsky’s instance on environmental and social impacts on intelligence (Shayer & Adey, 2002).

**Cognitive acceleration in practice.** As an intervention, Shayer and Adey (2002) built cognitive acceleration on six pillars of practice: (a) establishing the schema, (b) using concrete preparation, (c) introducing cognitive conflict, (d) applying social construction, (e) engaging in metacognition, and (f) bridging to other learning. In the first stage, a teacher prepares students for new learning by establishing the schema of understanding and preparing students to attach the upcoming content to prior knowledge.
The second pillar involves the use of a concrete introductory set in the Piagetian sense of an example that is understandable in its basic form as presented without the need for additional interpretation. In the third stage, the teacher introduces some aspect of cognitive conflict by presenting new learning or an example that seems to conflict with the initial concrete example provided. Applying social construction in the third phase refers to the use of group discussion and discourse to tease out deeper understanding of new learning and allow students to build a collective resolution of the cognitive conflict. This is reinforced in the fifth step where the students are encouraged to reflect on their new learning in a metacognitive process that directly addresses what they learned, how they learned it, and why it mattered for them. This is then resolved in the final stage where the new learning is bridged, or connected, to other content or other lessons that have been taught in the classroom (Shayer & Adey, 2002).

During a classroom lesson, the teacher will move through the pillars starting with an introduction based on a concrete aspect and then moving in a cyclical pattern of cognitive conflict, social construction through collaborative discussions, and generalization of understandings that lead to new cognitive conflicts (Adey, 2008). Not only did the students who received the cognitive acceleration science intervention score statistically significant higher marks on immediate posttests on cognitive development and science ability, but they also continued to receive statistically significant better scores on later science exams as well as English and math exams (Adey, 2005). Based on this evidence, Adey (2005) claims that cognitive acceleration can be seen to have both a long-term and far-transfer impact on student learning.
**Implications for computer science.** The cognitive acceleration theory offers a potential lens for shaping the necessary adult learning to prepare elementary teachers to lead computer science instruction. Cognitive acceleration has been applied to general technology education at the middle school level in prior work by Hamaker and Backwell (2003) suggesting a possibility for additional consideration in this study. The underlying concepts from Piaget and Vygotsky that cognitive acceleration synthesized have prior use in computer science education research. In an Australian study of computer science educators, a Neo-Piagetian approach was used to classify exercises as either pre-operational, concrete operational, or formally operational in terms of tracing code (Gluga, Kay, Lister, Kleitman, & Kleitman, 2013). Tracing, or stepping through code, refers to a line-by-line execution of a computer program by hand. This is a useful technique for debugging and is seen by Lister (2011) as a gateway for formal operational thinking. Another study from New Zealand recommends a supportive instructional style for teachers as a way to extend the range of Vygotsky’s zone of proximal development as seen in individual students (Awbi, Whalley, & Philpott, 2015). These examples, along with other research on the use of theories from Piaget and Vygotsky in computer science (Anderson & Gegg-Harrison, 2013; Lister, 2011; Teague, Corney, Ahadi & Lister, 2013) suggested that the synthesis of these two fields in cognitive acceleration made it a valid and appropriate theory for use in examining teacher development around computer science.

**Statement of Purpose**

The purpose of this study was to investigate potential methods for increasing the confidence and competence of in-service elementary teachers with respect to computer
science in order to prepare them to meet the increasing demand for elementary level computer science instruction. The theoretical framework provided by cognitive acceleration suggested that an approach to instruction built from a concrete understanding and developed through social constructivism could have a positive impact on teacher confidence and developmental growth concurrent with student instruction (Adey, 2008). Specifically, the constructivist aspects of cognitive acceleration in relation to a play-based approach using physical board games as compared to digital apps were investigated.

**Research Questions**

The study was designed around two research questions related to the potential impact the interface modality of the instructional tool used during professional development and teaching could have on elementary teachers.

1. Is the confidence of elementary teachers in an initial computer science professional development interaction different depending on the use of an analog or digital teaching tool?

2. Is the competence of elementary teachers in an initial computer science professional development interaction different depending on the use of an analog or digital teaching tool?

**Potential Significance of the Study**

There is a growing push for computer science instruction, but in-service elementary teachers may not have the requisite confidence and competence to be effective leaders in this subject. Historical factors have led to a lack of computer science instruction and professional development within elementary schools (Gallup & Google,
In a pilot study of the New Zealand computer science curriculum, Duncan and Bell (2015) noted that the assigned elementary computer science teacher would have been unable to lead instruction without the intervention of the research team. Successful instruction is critical for elementary computer science. Duncan et al. (2014) noted that a lack of teacher confidence and competence around computer science can lead to negative impacts on students for future computer science instruction. A current gap in the literature around computer science professional development for elementary teachers (Menekse, 2015) supported the need for this study. Finally, the application of cognitive acceleration as a synthesis framework suggested a pathway to extend the current understanding and application of theories from Piaget and Vygotsky in computer science education literature.

**Definitions of Terms**

*Computer science* refers to the broad field of study consisting of specialized aspects of mathematics and philosophy (logic) related to defining ideas that can be computed. Computer science is often taught within the context of a specific computer programming language, except at the highest levels where it is an abstraction of math and logic.

*Computational thinking* is a subset of computer science distinguished by a more language agnostic approach to instruction that often does not even involve computer hardware. Grover and Pea (2013) suggested that much computer science instruction at the elementary level is being implemented through computational thinking.

*Interface modalities.* This study looked at the potential impact of different interface modalities. Specifically, it compared analog and digital interfaces.
is the technology that mediates a human’s interaction with a computer or computational process. In an analog case like a board game, the interaction is purely tangible with no digital components. Hybrid interfaces may include tangible interfaces that facilitate access to digital interactions. Digital graphical interfaces to computers are fully contained within the computer hardware and are sometimes engaged through a touchscreen.

**Chapter Summary**

Computational thinking was defined by Wing (2006) as a subset of computer science focusing on the underlying skills foundational to understanding programming but transferable to other disciplines. Computational thinking has become a prominent approach for teaching computer science in elementary classrooms. A lack of professional development and other historical factors have left in-service elementary teachers ill-prepared to meet rising demands for instruction in computational thinking at early grades. The theory of cognitive acceleration (Shayer & Aday, 2002) suggested a potential path for further investigation around the use of concrete examples and discourse as a way to bring about effective learning. This study looked at potential differences in teacher gains in terms of confidence and competence depending on the use of either analog or digital interfaces during professional development. A review of current literature is presented in Chapter 2. Chapter 3 contains the procedures for the experimental design. Results are shared in Chapter 4. Finally, implications of this study and recommendations based on a review of the results are presented in Chapter 5.
Chapter 2: Review of the Literature

Introduction and Purpose

There is a growing push for computer science instruction, but in-service elementary teachers may not have the requisite confidence and competence to be effective leaders in this subject. Historical factors have led to a lack of computer science instruction and professional development within elementary schools (Gallup & Google, 2016). In a pilot study of the New Zealand computer science curriculum, Duncan and Bell (2015) noted that the assigned elementary computer science teacher would have been unable to lead instruction without the intervention of the research team. As such, teacher preparation is a topic needing additional research and consideration. Specifically, this study examined two research questions around the potential impact of an analog or digital resource on the development of confidence and competence during professional development for elementary teachers.

In this review of existing literature, empirical studies were examined to establish recognized best practice for teaching computer science as well as potential methods to prepare teachers to lead computer science instruction. Despite some emerging research, there is still a gap in the literature around computer science professional development for elementary teachers (Menekse, 2015). As such, related studies from mathematics were reviewed to add understandings from a similar discipline that has also faced issues around confidence and anxiety (Geist, 2015). Empirical studies about play-based learning
and teaching commuter science through play and games were also included in this review.

**Selection criteria for studies.** Every effort was made to focus on peer-reviewed empirical studies published in notable journals. However, given rapid changes within the field of computer science instruction and emerging research looking at teacher perceptions, some of the empirical studies included in this review are taken from conference proceedings from sections of the Association for Computing Machinery. Conferences from the Special Interest Group on Computer Science Education held in October 2016 and March 2017 resulted in papers relevant to this review that have not had time to be published in journals. The March 2017 conference had an acceptance rate of 30.2% with five peer reviewers for each submission; a level of review that approaches the academic rigor of journal selection (SIGSCE, 2017).

**Methodological review.** A total of 32 empirical studies were reviewed. Of those, 14 used a quantitative method, 6 used a qualitative method, and 12 used mixed methods. Most of the quantitative research studies sought to compare the efficacy of two or more possible interventions and so involved correlative procedures and measures of relationships. For example, as will be seen later, Oliveira, Nicoletti, and del Val Cura (2014) examined correlations between computational thinking and other subjects in school. The mixed methods studies tended towards the same approach, but with added analysis and information from a qualitative review of the participants and their interactions. An example of this is Strawhacker and Bers (2015) whose comparison of learning based on modality of interface was further enriched by a qualitative analysis of children’s conversations during the learning. Despite being the least frequent, the
qualitative studies involved in this review tended to provide very rich understandings of the thoughts and motivations of teachers working to implement computational thinking. This was especially evident in Duncan et al.’s (2017) review of teachers self-rated confidence levels after teacher computational thinking lessons. In addition to the 32 empirical studies, an additional 19 resources including meta reviews, historical, and theoretical papers were also included to provide additional context and information.

**Computational Thinking in K-12**

Computational thinking is a subset of computer science focusing on the underlying skills foundational to understanding programming but transferable to other disciplines. The term, originally coined by Papert (1980), was reintroduced by Wing (2006) in a seminal paper. Wing (2006) positioned computational thinking as a skill for everyone to learn, stressing that it is more conceptual and cross-disciplinary than traditional computer programming. In a review of computational thinking in K-12, Grover and Pea (2013) found broad interest in the computational thinking both as an approach to teaching computer science as well as a general instructional component in schools to enhance learning across disciplines. The literature was analyzed to answer three questions. First, the potential impact of computational thinking on general performance in schools was considered. At issue is whether schools might see enough benefit to add a computational thinking component to their curriculum. Second, the ability of computational thinking to directly teach computer science was reviewed. Finally, implications of a computational thinking approach to computer science were considered with respect to the confidence and competence of the teachers involved.
**Computational thinking actualized in schools.** Multiple studies have correlated mastery of computational thinking with increased achievement in other K-12 disciplines as well as general mental abilities and problem-solving skills (Chen et al., 2017; Oliveira et al., 2014; Román-González, Pérez-González & Jiménez-Fernández, 2017). These results suggested that, as Wing (2006) proposed, instruction in the computational thinking aspects of computer science can benefit all students, not just those pursuing a career in computer programming. For example, mastery of the systems and critical thinking aspects of computational thinking were strongly correlated with overall student academic achievement in a study by Oliveira et al. (2014). Similarly, a study of fifth grade students in the United States found a medium effect of computational thinking on everyday reasoning (Chen et al., 2017). In a very large study of students in Spain, Román-González et al. (2017) showed correlations between computational thinking and general mental abilities. These three studies indicated that further investigation of computational thinking is warranted both as an independent subject and as an instructional approach for teaching computer science.

**Computational thinking and computer science.** Computational thinking has been highlighted as best practice for early computer science instruction in both theoretical and empirical papers (Duncan & Bell, 2015; Fletcher & Lu, 2009). Fletcher and Lu (2009) opined that computational thinking should be taught before students are ever introduced to computer programming. Starting computer science instruction with programming, Fletcher and Lu (2009) claimed, is “akin to teaching basic arithmetic alongside proof construction, and elementary reading and writing with linguistics and discourse analysis” (p. 24). Instead, Fletcher and Lu (2009) argued, an early introduction
to the foundational aspects of computational thinking would better prepare students for later success in computer science and programming. This approach was supported by empirical evidence including a pilot study of New Zealand’s computational thinking-based computer science curriculum (Duncan & Bell, 2015). In a study of 330, 11- and 12-year-old students, Duncan and Bell (2015) were surprised by the normal distribution of scores on a programming quiz after instruction using a computational thinking model rather than the bi-modal distribution often seen in other studies. This suggested that despite a large range of 9.1% to 90.9% on the quiz, the computational thinking approach reduced the number of extremely low scores by increasing programming abilities for more students (Duncan & Bell, 2015).

Similar results were found by Cetin (2016) in an experimental study of 56 Turkish pre-service teachers taking either a traditional computer science course teaching the C programming language or a more computational thinking focused course that used Scratch. Unlike the text-based C language, Scratch is a block-based language that lets users drag-and-drop pre-defined blocks of code to construct a program. Like the pilot study reported by Duncan and Bell (2015), and the recommendations from Fletcher and Lu (2009), the experimental group in Cetin’s (2016) study focused on underlying concepts found in programming and the application of those ideas within collaborative, constructivist activities. As compared to the traditional code-focused instruction of the control group, the experimental group scored significantly higher on both knowledge (M = 57.18 vs. M = 45.54) and application (M = 58.14 vs. M = 39.54) tests after the study (Cetin, 2016). As with the results reported by Duncan and Bell (2015), Cetin (2016) also found more homogenously grouped results with a lower standard deviation from the
mean in the experimental group (SD = 11.10) as compared to the control group (SD = 14.10). These results suggested that the computational thinking approach as theorized by Fletcher and Lu (2009) might have helped students gain a more well-rounded understanding of the language they were learning. In conclusion, the work of both Cetin (2016) and Duncan and Bell (2015) supported the efficacy of computational thinking as a way to teach computer science.

**Competence and Confidence in Computational Thinking**

There are additional reasons to adopt a computational thinking approach to teaching computer science at the elementary level beyond the established instructional efficacy of the method (Cetin, 2016; Duncan & Bell, 2015). The unique approach of computational thinking as a problem-solving, playful, engagement with computer science is also linked with increased competence and confidence (Chen et al., 2017; Curzon, McOwan, Plant, & Meagher, 2014; Duncan et al., 2017; Lambert & Guiffre, 2009; Oliveira et al., 2014; Román-González et al., 2017). Competence and confidence are regularly used as defining measures of success within computer science (Curzon et al., 2014; Duncan & Bell, 2015) and so needed further consideration in this review.

**Competence and computational thinking.** A strong correlation between computational thinking mastery and academic success on tests in other school subjects supported increased instruction using a computational thinking approach (Oliveira et al., 2014). In a review of test scores from 81, 11- to 15-year-old students in Brazil, Oliveira et al. (2014) used Pearson’s product-moment correlation to show strong and moderate correlations between a student’s score on a test of computational thinking and grades in other core academic subjects. As instruction and mastery of computational thinking
increased, so did student achievement in general across other academic disciplines. The systematic approach and problem-solving focus of computational thinking might be the common factor that explains the correlation with increased learning in other subjects, Oliveira et al. (2014) proposed, but further study would be needed to establish causality.

Supporting the assertions of Oliveira et al. (2014) regarding the commonality of problem solving and reasoning between computational thinking and other fields, Chen et al. (2017) also found a significant correlation. In a small study of 5th-grade students in the United States, Chen et al. (2017) used a multiple paired t-test to show a significant medium effect (.69) for computational thinking on everyday reasoning. This, Chen et al. (2017), concluded, showed some evidence of transfer of learning from the computational thinking-based robotics instruction in the study to other aspects of problem solving. Like Oliveira et al. (2014), Chen et al. (2017) suggested that further research was necessary to examine the nature of the transfer and the potential for increasing transfer through instructional techniques that more explicitly connect computational thinking to other problem-solving scenarios.

In a large study of over 1,200 5th- through 10th-grade students in Spain, computational thinking was linked to increased mental abilities as well as increased general problem-solving (Román-González et al., 2017). Scores from the Primary Mental Abilities tests for verbal, spatial, and reasoning skills were found to have a positive correlation with the validated Computational Thinking Test. Further regression analysis by Román-González et al. (2017) revealed that the spatial and reasoning tests specifically explained 27% of Computational Thinking Test scores. While the spatial and reasoning results might be expected given the emphasis on puzzle and problem solving in
computational thinking, the positive correlation with verbal abilities might seem a bit odd at first, especially considering that there was no significant correlation between computational thinking and the numerical abilities test. Given the focus in Wing’s (2006) explanation of computational thinking as conceptualizing and describing as opposed to creating computational artifacts, a correlation with verbal abilities should not be unexpected. Román-González et al. (2017) described the verbal test as evaluating a student’s capacity to “understand and express ideas with words” which is seen in the highly language-based block language used within the Computational Thinking test.

These three studies suggested that computational thinking has potential applications both as an assessment or predictor of a student’s general academic performance, and also as an instructional practice to increase student achievement across all subjects (Chen et al., 2017; Oliveira et al., 2014; Román-González et al., 2017). They validated Wing’s (2006) call for computational thinking to join reading, writing, and arithmetic as a new core subject in elementary schools. The three studies also provided some answers to questions raised by Grover and Pea (2013) on the potential of computational thinking to be intentionally implemented to transfer problem solving and thinking skills into other subjects. Finally, given a correlation with overall academic success for students, the research suggested that implementing instruction on computational thinking could be beneficial for all elementary schools (Oliveira et al., 2014).

Confidence and computational thinking. Using a computational thinking approach to computer science has also been related to confidence for teachers and students in a number of studies (Curzon et al., 2014; Duncan et al., 2017; Lambert &
In a small study involving pre-service teachers at a Virginia university working with elementary students using computational thinking activities, Lambert & Guiffre (2009) reported significant increases in confidence measures about both computer science and math. Lambert and Guiffre (2009) introduced computational thinking using CS Unplugged activities developed in part by Tim Bell of New Zealand who has been involved in many of the other studies discussed here. For example, Duncan et al. (2017) provided updated teacher perceptions on the ongoing implementation of the New Zealand computer science curriculum using the same CS Unplugged activities as used by Lambert and Guiffre (2009). Overall, most of the 13 teachers reviewed by Duncan et al. (2017) self-reported as being moderately or very confident; only three teachers indicated some level of unconfident feelings. When the instruction began, however, most of the teachers self-reported as having very low confidence suggesting that success during the implementation of a computational thinking-based approach to computer science may have inspired greater confidence (Duncan et al., 2017). This possibility is also partially supported by Curzon et al. (2014) who reported on survey results from teachers attending workshops on computational thinking where most respondents (89%) agreed that the workshop increased their confidence. Given these general benefits, the question then turned to the specific use of computational thinking to teach computer science.

**Summary of computational thinking literature.** The studies reviewed above, along with additional meta-reviews of the existing literature, all supported the efficacy of a computational thinking approach to teaching computer science in elementary schools (Grover & Pea, 2013). By focusing on the underlying skills, foundational to computer science and programming but transferable to other subjects, a computational thinking
A person’s instinct for play can be leveraged within a carefully designed environment that melds play and learning through constructivist interactions (Rieber, 1996). In his seminal article that redefined the modern concept of play-based learning, Rieber (1996) suggested the use of microworlds to frame constructivist learning within an environment of playful exploration. In this instance, a microworld is defined as an intentionally created experience with a specific focus that guides play. The approach suggested by Rieber (1996) emphasized engagement and intrinsic motivation through mindful attention to learning about content through play. More recently, play has been described in terms of approaches to problem solving and project-based learning (Thorsted, Bing, & Kristensen, 2015). Like the microworlds approach described by Rieber (1996), Thorsted et al. (2015) approached project-based learning as a type of constructivist learning that could be enriched by the exploration, wonder, and drive to solve that is inherent to play. To differentiate the approach taken in this study, Thorsted et al. (2015) referenced the German tradition of Bildung – a process of self-creation that implies the development of both confidence and competence (Nordenbo, 2002). Because of the play-based approach, Thorsted et al. (2015) observed that the students involved gained confidence and took on new approaches to learning that creatively integrated
explicit and tacit prior-knowledge with the learning of the moment. These outcomes were closely aligned with many aspects of computational thinking as a new way of approaching problems and problem solving and suggested the potential for a play-based approach to computational thinking instruction.

Increasing confidence for computational thinking with play. A play-based approach to teaching computational thinking and computer science can increase confidence and provide a strong base for learning (Duncan & Bell, 2015; Mathrani, Christian, & Ponder-Sutton, 2016). In the pilot study by Duncan and Bell (2015) referenced above, the teacher wanted more time to play with Scratch “to help build confidence so that everyone feels comfortable using Scratch before trying to add in programming concepts” (p. 8). The informal nature of play established a baseline of experience and comfort with new technologies prior to formal instruction. Anxiety is commonly seen in teachers concerning the use of technology (Downey & Kher, 2015; Efe, 2016; Sanalan, 2016) so instructional approaches that reduce anxiety and increase confidence could help improve the efficacy of teachers. Play has been shown to accomplish this in the related field of mathematics instruction (Cohrsen, Church, & Tayler, 2016; Cohrsen, Tayler & Cloney, 2015) where a play-based approach increased teachers’ confidence in their math abilities.

Implications of a lack of confidence. Like computer science and technology use in general (Downey & Kher, 2015; Efe, 2016; Sanalan, 2016), mathematics is a subject that is often viewed as challenging by elementary teachers (Chang & Beilock, 2016; Geist, 2015). This is important because as Geist (2015) found, anxiety towards mathematics was related to a teacher’s reported ability in math which in turn was linked
to the importance placed on teaching math and the amount of instructional time devoted to math. In other words, teachers with math anxiety taught math less than math-confident teachers (Giest, 2015). Similarly, Turkish pre-service teachers with higher levels of anxiety about technology were found to use technology less within their student teaching placements (Aslan & Zhu, 2016). If this holds true for computer science, then addressing teachers’ confidence will be a critical component of implementing computer science instruction.

**Play increases confidence in computer science.** The confidence gained through play establishes a more productive learning environment for initial instruction or as a therapy to modify preexisting negative impressions of computer programming (Mathrani et al. 2016). Participants in the game-based PlayIT program developed by Mathrani et al. (2016) reported higher levels of confidence from playing the programming game and showed increased passing rates on an end-of-course programming exam. PlayIT was implemented at an independent, non-university, training center in New Zealand based on structured national learning standards and certification exams. As such, this study provided an opportunity to explore the role of play as remediation for past negative learning experiences as well as a new approach for first-time students of computer programming. Students from the two cohorts that used the programming game had much higher passing rates on the exam as compared to the non-participating cohort (Mathrani et al., 2016). Mathrani et al. (2016) found that 86% of the game-based learning cohorts passed the certification exam on their first attempt of an allowed three tries as compared to only 44% of the control cohort. Furthermore, only one student of the 44 in the experimental cohorts failed after three attempts compared to five of the 27 students in the
control cohort. Game-based learning was an effective instructional strategy for teaching computer programming, but perhaps more importantly, the increased content mastery was also accompanied by increased confidence in programming ability.

**Teaching through play impacts teacher confidence.** The impact of teaching through play on teachers’ confidence with the subject being taught has been seen in other subjects (Cohrssen et al., 2016; Cohrssen et al., 2015). Though there have been no studies examining a game or play-based approach to professional development for in-service teachers, related studies from mathematics have been conducted. One implementation of a play-based mathematics program with Australian early-childhood teachers was reported in both quantitative (Cohrssen et al., 2015) and qualitative (Cohrssen et al., 2016) studies. From a quantitative perspective, Cohrssen et al. (2015) used regression analysis to show a significant relationship between the teachers’ use of the play-based interventions and student learning. Additionally, the teachers who used the play-based intervention showed increased competence as measured by the use of proper mathematical language in describing models (Cohrssen et al., 2015).

Qualitatively, the teachers also reported higher levels of personal confidence with mathematics following the intervention (Cohrssen et al., 2016). Cohrssen et al. (2016) focused on the transformation of one teacher who revealed at the start of the study that her personal experience with mathematics was quite negative: “I don’t know what everyone else’s experience is, for me it would be about I just shut down so I wouldn’t think at all” (p. 8). By the end of the study, the same teacher was much more self-aware: “I don’t have that confidence in mathematics as well and I think that’s something I need to work on because I’m trying to give something to the next generation, to give them the
groundwork and the interest in maths, not just literacy” (Cohrssen et al., 2016, p. 8). After implementing the play-based approach to mathematics instruction, the teacher was going to keep using the program, recognizing that by teaching math in this way she was also learning math herself (Cohrssen et al., 2016). Though these qualitative results cannot be generalized to other situations, they nonetheless illustrate a single point of transformation supporting the need for further study within computer science instruction on the role of play.

**Summary of computational thinking confidence literature.** Given the importance of confidence with a subject as a factor of instructional time spent on a subject, it is important that teachers be presented with professional development and instructional approaches that build their confidence (Cohrssen et al., 2016; Giest, 2015). A play-based approach was seen to be effective in multiple empirical studies in computer science and related fields (Cohrssen et al., 2016; Duncan & Bell, 2015; Mathrani et al., 2016). Play can take on many forms, however, so additional consideration must be given to the interface modality that provides the strongest support for developing confidence and competence in teachers as leaders of computational thinking instruction.

**Interface Modalities and Playful Approaches to Computational Thinking**

One aspect of play that can be shown to have a relationship with increased competence and confidence in computer science is the modality of the interface used during the interaction (Horn, Crouser, & Bers, 2012; Horn, Solovey, Crouser, & Jacob 2009; Kim et al., 2015; Strawhacker & Bers, 2015; Wohl, Porter, & Clinch, 2015). In most cases, the studies were looking at differences between a digital user interface on a computer and an analog user interface including either purely physical components
(unplugged) or a combination of analog and digital components (hybrid). In a comparison of kindergarten interactions and the resulting mastery of programming concepts depending on the use of an analog, hybrid, or digital interface, Strawhacker and Bers (2015) found significant increases in learning for the analog group but not for the other interface groups.

**The impact of interface modality on competence.** The use of a analog or hybrid interface has been correlated with increased mastery of computer science concepts in several studies (Horn et al., 2012; Strawhacker & Bers, 2015; Wohl et al., 2015). For example, Horn et al. (2012) observed interactions of children at a science museum and in kindergarten classrooms to compare interactions between analog and digital interfaces. For the children involved, learning was similar regardless of the interface modality, however Horn et al. (2012) do note that the analog interface was more productive for group instruction settings. This finding is supported in a later study by Strawhacker and Bers (2015) where analog interfaces were shown to have the highest student gains in another kindergarten situation. In pre/post test score comparisons across classrooms teaching with either an analog, digital, or hybrid user interface the only significant gains were seen in the analog interface classroom (Strawhacker & Bers, 2015). Strawhacker and Bers (2015) also reported a significant difference in posttest scores on one activity where the analog group outperformed the hybrid group. Despite these few significant findings, Strawhacker and Bers (2015) do note that overall scores were very tightly grouped between all three interfaces. Similarly, Horn et al. (2012) concluded that all types of interfaces serve a purpose in teaching computer science; digital interfaces were
especially useful for individual, self-guided, student work, but analog or hybrid interfaces were best suited for teacher-led instruction.

In terms of instructional gains, another study comparing analog, hybrid, and digital interfaces found the highest mastery of concepts from the use of analog resources (Wohl et al., 2015). In a qualitative study of five- to seven-year-old students in rural schools in the United Kingdom, Wohl et al. (2015) looked at differences between unplugged resources (analog), Cubelets robots (hybrid), and the Scratch programming language (digital). This study was unique in that it sought to reveal potential differences resulting from the order of introduction when all three modalities were used as well as different learning from each modality of interface. Most notably, Wohl et al. (2015) found that the responses to interview questions greatly depended on the modality of the resource used in that day’s instruction. On days when students used the digital interface of Scratch, answers were more focused on the application of the tool while answers after using the analog unplugged resources answers were more rooted in conceptual understandings (Wohl et al., 2015). The sessions using the hybrid interface provided by Cubelets robots elicited both tool-based and concept-based answers (Wohl et al., 2015). Unsurprisingly, Wohl et al. (2015) found that the analog unplugged resources were the most effective at increasing student competence around the concepts of computer science as commonly seen in computational thinking instruction (Wohl et al., 2015). This suggested that teachers might be able to influence the focus of learning and understanding by modifying the interface modality used for instruction. For mastery of computational thinking, an analog interface may be best suited.
The identification of different learning styles emerging from different interfaces was also seen in a study of physical and virtual robotics instruction by Berland and Wilensky (2015). This study compared analog and digital interfaces at two Chicago middle schools. Overall, both groups showed improvement in their understanding of robotics regardless of the interface; on some posttest questions the analog group scored significantly higher and on others the digital group did (Berland & Wilensky, 2015). There were subtle differences in the learning, though, that went beyond test scores. Berland and Wilensky (2015) reported that the analog interface learners created robot circuits from an “agent-based perspective” while the digital interface learners designed “aggregate perspective” systems. This was evidenced by the analog interface group creating robots designed for independent movement and action with a much stronger focus on the agency of the robot as an almost living being even when this resulted in a less efficient circuit (Berland & Wilensky, 2015). Berland and Wilensky (2015) posited that the digital group could more rapidly create prototypes within the virtual environment and so could adopt a higher perspective on problem solving while the analog group focused on interplay between and within the system. These findings echoed the results from Wohl et al. (2015) showing that different interfaces lead to different types of learning.

The impact of interface on competence is not limited to children; even adults have been shown to perform more competently using analog interfaces compared to digital (Schneider, Jermann, Zufferey, & Dillenbourg, 2011). In a study of 82 Swiss logistics apprentices aged 16 to 40, Schneider et al. (2011) found that the apprentices who used an analog interface for laying out shelving in a warehouse performed better than the group
that used a digital interface. The analog group placed more shelves in the warehouse \( (M = 32.2, \ SD = 2.6) \) compared to the digital group \( (M = 27.1, \ SD = 3.8) \) and also had better access from the shelves to the docks (Schneider et al., 2011). In this study, the analog group was literally playing with blocks, the shelving layout was accomplished by placing small wooden shelving units on a grid (Schneider et al., 2011). The introduction of playfulness through an analog interface resulted in additional differences in the interactions.

**The impact of interface modality on confidence.** Interface modalities can be seen to have significant impacts on confidence as observed through interaction preferences, level of engagement, and playfulness (Fails et al., 2005; Horn et al., 2009; Schneider et al., 2011). In a seminal article comparing analog and graphical interfaces for play, Fails et al. (2005) found greater levels of engagement and interest during the analog situation as compared to the graphical. As established above, within the similar field of mathematics engagement and interest are indicators of overall confidence with the field (Chang & Beilock, 2016; Geist, 2015). Additional studies have explored the questions raised by Fails et al. (2005).

In one such study, the interface modality of an exhibit at a science museum was shown to result in different levels of interaction and engagement (Horn et al., 2009). Perhaps most notable about the findings from Horn et al.’s (2009) observations of children and adults presented with either an analog or digital interface was a significantly higher preference for the analog interface based on gender. Overall, on days that the
Figure 2.1. Percentage of visitors interacting with an exhibit based on a graphical user interface (GUI) or a tangible user interface (TUI) by gender. Reprinted from “Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education” M. S. Horn, et al., 2009, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, p. 980. Copyright 2009 by the Association for Computing Machinery. Reprinted with permission.

exhibit featured an analog interface, a significantly higher percentage of users stopped to engage; 51.3% of visitors for the analog as compared to 30.6% of visitors for the digital interface as shown in Figure 2.1 (Horn et al., 2009). When disaggregated for gender, the differences became much more notable. Girls were almost three times as likely to interact with the analog exhibit while boys were only about twice as likely to use the analog exhibit (Horn et al., 2009). Women were also more than twice as likely to stop and engage with the analog interface as compared to the digital interface (Horn et al., 2009). A more recent study by Sapounidis and Demetriadis (2013) had similar results regarding increased first sight preference and higher levels of engagement with a analog interface
as compared to a digital interface. This suggests that when introducing teachers to computer science, an analog interface might be more effective during initial interactions to build confidence and encourage initial engagement.

Using an analog interface can also change the nature of the interaction to be more creative, collaborative, and playful (Schneider et al., 2011; Xie, Antle, & Motamedi, 2008). Comparing the mediatory values in the warehouse shelving study addressed above, Schneider et al. (2011) revealed that the analog group explored more options, collaborated and discussed options more, and had a higher level of playfulness as measured by the concept of flow. Perhaps more importantly, the deeper level of engagement and interaction by the analog group hypostasized as being a factor in the increased learning demonstrated by that group (Schneider et al., 2011). A failure to properly test the pre/post test questions, noted Schneider et al. (2011), resulted in a posttest question on warehouse layout that was significantly harder. This mistake in the design of the study, Schneider et al. (2011) proposed, accounted for the reduced scores amongst the digital group between pre/post testing and made the significantly increased scores of the analog group that much more notable. Given the strong focus of computational thinking on solving problems, the use of an analog interface that can possibly increase exploration, collaboration, and playful engagement has great potential (Schneider et al., 2011).

**Summary of interface modality literature.** As was seen in the studies discussed above, the interface modality for the resource used to teach computational thinking or computer science matters. Schneider et al. (2011) provided strong evidence to support increased learning and results – strong indicators of competence – from the use of an
analog interface as compared to a digital interface. Other studies suggest that despite some evidence for analog preferences, hybrid and digital interfaces still play an important role depending on the instructional objectives and resources used (Horn et al., 2012; Strawhacker & Bers, 2015). However, Horn et al.’s (2009) observations of interactions with analog or digital interfaces at a museum supported the preference for an analog interface not only in children, but also for adult women. With women making up the majority of elementary teachers, this indicator of a more than two to one confidence preference for an analog interface cannot be ignored (Horn et al., 2009). Analog interfaces were therefore seen to increase both competence and confidence as well as increase engagement within a playful environment (Horn et al., 2009; Schneider et al., 2011). One way to maximize the potential benefits of these aspects of analog interfaces is using board games.

**Board Games as Analog, Playful Computational Thinking**

Games, particularly analog games like board and card games, provide a way to implement play-based instruction in a classroom to leverage the findings about analog interfaces and playful engagement discussed above. There are, however, challenges to using games for learning including alignment of the game to learning objectives and pedagogical and cognitive appropriateness of the gameplay (Koh, Kin, Wadhwa, & Lin, 2012; Phillips, Horstman, Vye, & Bransford, 2014). In a study of Singaporean teachers’ perceptions about the instructional use of games, Koh et al. (2012) found that after the expected barriers to implementing any new program – a lack of time, limited resources, and high costs – the biggest concern expressed was for alignment of the game to classroom teaching objectives. Phillips et al. (2014) also addressed concerns regarding
the pedagogical approaches in games. In a study of a math game, Phillips et al. (2014) examined two case studies in which the game failed as an instructional approach. In one case, the math in the game was too easy resulting in disengagement by the student while in the second case the student’s use of repeated trial-and-error problem solving techniques could appear successful while not actually resulting in new learning (Phillips et al., 2014). These concerns are valid in considering the potential use of games both as an instructional approach for teaching computational thinking and as a way to build teacher competence and confidence with the subject of computational thinking.

**Games as an instructional approach for computational thinking.** Games can serve as an effective instructional medium for embedded instruction of computational thinking and computer science in both analog and digital modalities (Berland & Duncan, 2016; Berland & Lee, 2011; Hsu, Tsai, Chang, & Liang, 2017). In a theoretical article, Apostoellis, Stewart, Frisina, and Kafura (2014) describe an instructional board game called RaBit EscAPE designed to specifically address the dispositions of computational thinking. Others have aligned existing commercial board games to describe specific computational thinking skills empirically observed during gameplay (Berland & Duncan, 2016; Berland & Lee, 2011). Despite the different approaches focusing on either skills or dispositions in gameplay, the resulting understandings about student interactions with computational thinking are quite similar. For example, the skill of distributed computation requires mastery of the ability to work with others and establish a collaborative environment to distribute work effectively (Berland & Lee, 2011). Teachers who are new to computational thinking might be more confident using a game that
introduces dispositions before trying to tackle a more competence-focused, skills-based
game.

Empirical evidence for the use of computational thinking as a part of playing
commercial board games was established in a seminal article by Berland and Lee (2011).
Prior to this, studies had mostly focused on the use of digital computer games to teach
computer science (Papastergiou, 2009). Berland and Lee (2011) observed college
students as they played *Pandemic*, a cooperative board game in which players work to
cure diseases around the world using set collection and resource allocation mechanisms.
The act of playing a collaborative board game, Berland and Lee (2011) posit, resulted in
verbal interactions between players that made players’ motives and use of skills more
observable. Berland and Lee (2011) suggested two aspects of board game use that
contributed to the efficacy of the study. First, that the collaborative nature of *Pandemic*
encouraged open discussion and sharing, and second that the physical nature of a board
game required the players to take on roles that would have been relegated to the computer
in a digital game (Berland & Lee, 2011). This aspect of collaborative board games is also
supported by other studies; Peppler, Danish, and Phelps (2013) also found increased
interaction and group learning using a collaborative play style as compared to a
competitive version of the same game.

Another critical aspect of the classroom use of board games is that the physical
medium allows for manipulation by the teacher to create intentional instructional
interactions (Harris & Harris, 2015). Berland and Duncan (2016) manipulated the rules of
*Pandemic* as a follow-up to the initial study from Berland and Lee (2011) discussed
above. In the newer study, Berland and Duncan (2016) compared the computational
thinking interactions between players in an unmodified version of the game and those using three different modifications. The modified interactions included a player cheat sheet that simplified and clarified the rules, a ghost player that the other players collaboratively managed, and a changed set of rules that highlighted comparisons to real diseases (Berland & Duncan, 2016). After analysis, the only significant differences were an increased application of global and local logic using the modification to the rules that had players collaborate to control an additional ghost player (Berland & Duncan, 2016). This finding also supports the findings of Peppler et al. (2013) regarding collaborative play referenced above.

**Building competence and confidence through board games.** Compared to traditional competitive games, collaborative board games can inspire more content learning, player engagement, and positive interactions (Peppler et al., 2013). These three aspects can also be interpreted as indicators of competence and confidence. Peppler et al. (2013) observed significantly more conversation that used scientific vocabulary and addressed the specific science content of the game during collaborative play as compared to competitive play. This suggests that the collaborative version of the game board at least provided players with more opportunities to demonstrate competence and mastery of the content involved. Furthermore, Peppler et al.’s (2013) finding that collaborative game play resulted in more positive interactions (22 vs. 7) and fewer negative interactions (2 vs. 30) than competitive play also suggests an environment more conducive for building confidence. In another measure associated with confidence, players in the cooperative version of the game were also significantly more engaged as compared to players in the competitive version (Peppler et al., 2013).
A unique aspect of games that can build both confidence and competence is the drive to overcome a challenge and complete a task (Phillips et al., 2014). This is perhaps best summarized in the definition of a game from philosopher Bernard Suits (2005) who stated that “playing a game is the voluntary pursuit to overcome unnecessary obstacles” (p. 55). Phillips et al. (2014) found strong support for this definition in students who reported their emotional response to playing a math game as both positive and frustrating. “I got frustrated,” reported one student, “but it was like a good kind of frustrated” (Phillips et al., 2014, p. 557). The frustration felt by many of the students drove them to complete the game to overcome a personal challenge (Phillips et al., 2014). Teachers could make use of this drive to engage students in gameplay as a form of learning but must be careful not to become overly focused on the game itself. “Using game analytics alone as the guideposts for designing games for learning may produce game play experiences that are detrimental to learning” Phillips et al. (2014) warned (p. 563).

Board games have also been developed in recent years that explicitly teach computational thinking and computer science within an analog environment (Geist, 2016). One of the most widely used games is Robot Turtles (Shapiro, 2014), a board game for young children that replicated the turtle-based programming environment found in the LOGO language. Despite being a board game, Robot Turtles included complex
programming concepts including subroutines and callable functions that pushed children to think at higher levels as seen in Figure 2.2 (Geist, 2016). The conceptual basis of Robot Turtles was also extended by an Indian game, Haathi Mera Saathi (My Elephant Friend) that also introduced programming to young children (Unnikrishnan, Amrita, Muir, & Rao, 2016). In Haathi Mera Saathi, the game space was extended from tangible

Figure 2.2. Directions from Robot Turtles showing complex computer programming concepts in a board game intended for young children. Reprinted from Robot Turtles (p. 12) by D. Shapiro, 2014, Alexandria, VA: Think Fun. Copyright 2014 by Think Fun. Reprinted with permission.
to a fully embodied experience where children were programed to move through a sequence (Unnikrishnan et al. 2016).

**Teacher approaches to play.** Selection or design of the game to be used for teaching computational thinking is critical as it must provide both an authentic play experience as well as a pedagogically appropriate instructional approach (Phillips et al., 2014). The teacher is required to have a certain level of expertise regarding both games and computer science during the selection process prior to implementation. Hsu et al. (2017) also highlighted the importance of this dual competence in both game knowledge and computer science knowledge required to make this instructional approach work. In a study of 316 Taiwanese in-service teachers using digital games, Hsu et al. (2017) found that a teacher’s concept of game-based pedagogical content knowledge was significantly predicted by that teacher’s motivation, confidence, and knowledge of games in general. The study found no significant prediction based on age, gender, or teaching experience for a teacher’s ability to implement game-based technology instruction (Hsu et al., 2017).

Games provided a mediating factor that leveled the field for all teachers to be able to implement technology instruction based on the teachers pre-existing levels of confidence, motivation, and general games knowledge. This was supported by a review of a Canadian professional development course for teachers about games and gaming (Becker, 2007). Becker’s (2007) course introduced teachers to game literacy and the background games knowledge called for by Hsu et al. (2017). Becker (2007) noted that at the start of the course, “most of the participants had imagined that digital games in class would be used as independent study aids, or something that, if it did not actually threaten their jobs, was to be used without much input from them” (p. 484). This notion of passive
game play is unique to digital games; Berland and Duncan’s (2016) entire study of board game usage was built around the expectation of active teacher involvement and even teacher manipulation of the game.

Summary of game-based teaching literature. As a highly interactive medium, games have a unique ability to engage players within an environment that is both playful and challenging (Phillips et al., 2014). Within a game space, players can be pushed to work harder and complete more difficult tasks than might be expected during other interactions (Phillips et al., 2014). This seems to be especially true of cooperative board games where the medium and the collaborative nature of play encourage social interaction and constructivist learning (Berland & Duncan, 2016; Berland & Lee, 2011; Peppler et al., 2013). Given these promising results from studies involving students, more attention needs to be given to the impact of game-based instruction on the teachers.

Gaps and Further Research

A meta review of studies from 2004 to 2014 about professional development for teachers on computer science revealed a clear gap in the literature around elementary education (Menekse, 2015). Menekse (2015) identified 21 studies on K-12 computer science professional development, yet there were no studies that specifically addressed the professional development needs of elementary teachers. More recently, some studies including elementary teachers have started to emerge. Pollock et al., (2017) reported at the March 2017 Special Interest Group for Computer Science Education conference on findings from a computer science professional development program in Delaware. In this study of 28 teachers, there were 5 elementary teachers (Pollock et al., 2017). The small
sample size, however, did not allow for disaggregation by level taught and so the study does little to inform on the needs of elementary teachers.

This is a startling gap in the literature given widespread recommendations that computer science and computational thinking should start as early as possible (Fletcher & Lu, 2009; Lee et al., 2011). While there are many studies that investigate how students in primary and elementary students react to computer science instruction, only recently has any attention been paid to the teachers as a source of ongoing instruction after the study (Duncan et al., 2017). The study by Duncan et al. (2017) is especially notable here in that it follows on from an original pilot study from Duncan and Bell (2015) in which it was observed that the teacher would have been unable to lead instruction without the support of the research team.

**Teacher preparation for computational thinking.** Despite an almost total gap in current empirical research focused on elementary teachers with respect to computational thinking and computer science, some theoretical recommendations have been advanced. Angeli et al. (2016) considered the implications for teacher preparation given a specific computational thinking framework of instruction and concluded that teachers needed to learn both technical content knowledge and specific pedagogical approaches that would let them most effectively teach the technical content. Despite a lack of explicit teacher preparation ideas, Angeli et al. (2016) did note that creating models was effective but required direct instruction and a great deal of support for the teacher-learners. Other best practices for teacher preparation can be extracted from the empirical studies reviewed before. Specifically, practices including the use of analog or hybrid interfaces (Berland & Wilensky, 2015; Schneider et al., 2011; Strawhacker &
Bers, 2015), the use of a play-based approach to learning (Mathrani et al., 2016), and the use of board games for computational thinking (Berland & Lee, 2011) were seen to be effective and worthy of further research.

Future research on interface modality. Existing literature provided a strong foundation for understanding the relationship between interface modality and learning (Berland & Wilensky, 2015; Schneider et al., 2011; Strawhacker & Bers, 2015). In some cases, an analog interface was shown to be more effective than a graphical interface (Schneider et al., 2011; Strawhacker & Bers, 2015) while other studies revealed similar gains or distinct differences in the type of learning depending on the modality of the interface (Berland & Wilensky, 2015). However, as the target population for this study was majority female, the initial preference for an analog interface by gender as observed by Horn et al. (2009) indicated that the most attention be given to analog interfaces. What was specifically lacking from the current literature was an examination of the potential for differences in gains regarding confidence and competence in computational thinking that could be realized from different interface modalities for in-service teachers. Studies have been conducted for other populations, but not for in-service elementary teachers.

Future research on play-based learning and computational thinking. There has been a great deal of existing research on play-based learning in elementary classrooms and teachers’ perceptions of play (Cohrssen et al., 2016; Cohrssen, et al., 2015; Thorsted et al., 2015). Additional research was needed to extend the literature around the use of play-based approaches to computational thinking in elementary classrooms. Mathrani et al. (2016) showed the efficacy of a play-based approach to computer science instruction as compared to a traditional approach within a non-school
population of learners. This study was designed to extend the literature regarding the use of a playful approach to computational thinking instruction.

**Future research on board games for computational thinking.** There has been some research on the use of games, especially board games, as a pedagogical implementation of play-based computational thinking using an analog interface (Berland & Lee, 2011; Berland & Duncan, 2016). Cooperative board games have been shown in multiple studies to be an effective way of encouraging the use of computational thinking skills in a collaborative environment built around socially constructed learning (Berland & Lee, 2011; Berland & Duncan, 2016). Peppler et al. (2013) also showed the efficacy of cooperative board games for encouraging a higher level of engagement, on-task behavior, and content-specific vocabulary as compared to traditional competitive games. Again, what was missing, was research on the impact of these games on teachers’ perceptions of confidence and competence with the subject of computational thinking. Hsu et al. (2017) found that elementary teachers in Taiwan tended to have positive perceptions of confidence and competence around the use of games for teaching technology. Furthermore, the perceptions regarding the use of games for teaching technology remained positive independent of age, gender, and years of experience, suggesting that this was an intervention that could be broadly accepted amongst an elementary teacher population (Hsu et al., 2017). Yet this research addressed digital games and was conducted in a country that had recently engaged in a national push for games-based teaching (Hsu et al., 2017). Given the established efficacy of analog interfaces seen above, additional research was needed to examine potential adoption of board games for use by teachers in the United States. Games like *Robot Turtles* and *Haathi Mera Saathi*
provided examples for analog learning experience that explicitly taught computational thinking and programming concepts but required additional empirical study to be validated as effective resources.

**Chapter Summary**

There has been a growing push for computer science instruction, but in-service elementary teachers may not have the requisite confidence and competence to be effective leaders in this subject. A significant gap existed in the literature regarding effective approaches to professional development for elementary teachers around computer science. This review of the existing literature suggested three key areas for further study on ways to increase confidence and competence in teachers as leaders of computer science instruction. First, a professional development approach based on computational thinking could help introduce computer science within a more constructivist environment. This was shown to support gains in confidence and competence for students. Second, students and pre-service teachers responded positively to play-based instruction around computational thinking, and preschool math teachers reported increased confidence and competence after implementing a play-based instructional approach. This suggested a need for more research on the use of play for computational thinking professional development. Third, analog interfaces provided increased gains in confidence and competence as compared to digital interfaces for students and adult learners with an additional gender preference for analog interfaces exhibited by girls and female adults. Based on these three aspects, support was seen for a study looking at potential differences between the use of analog and digital play-based interactions for computational thinking professional development.
Chapter 3: Research Design Methodology

Introduction

Given a lack of research regarding professional development for in-service teachers on computer science and computational thinking, this study sought to provide foundational understanding around possible approaches to professional development. Based on existing literature, one area that showed promise for additional research was the potential impact that the interface modality of instructional resources may have on the confidence and competence of the teachers involved (Berland & Lee, 2011; Duncan et al., 2017; Fletcher & Lu, 2009). In some studies, analog or tangible interfaces were seen to increase both competence and confidence as well as increase engagement within a playful environment (Horn et al., 2009; Schneider et al., 2011). Other studies suggested that despite evidence for analog preferences in children and adults, digital interfaces may still play an important role depending on the instructional objectives (Horn et al., 2012; Strawhacker & Bers, 2015).

Based on gaps in existing literature, this study investigated potential differences in elementary teachers’ confidence and competence depending on the use of either analog or digital instructional resources during professional development. Specifically, two research questions were investigated in a qualitative study.

1. Is the confidence of elementary teachers in an initial computer science professional development interaction different depending on the use of an analog or digital teaching tool?
2. Is the competence of elementary teachers in an initial computer science professional development interaction different depending on the use of an analog or digital teaching tool?

Research Design

This study was conducted as a qualitative, single-case experimental design using a non-concurrent, multiple-baseline approach with comparisons across subjects. Single-case experimental design is recognized in the literature as an experimental approach appropriate for use with smaller population sizes when studying larger groups is not feasible (Hitchcock, Kratochwill, & Chezan, 2015; Smith, 2012). In this instance, a lack of prior research made calculating an effect size and a requisite sample size for a traditional experimental design difficult. Single-case designs can still be used to establish a causal relationship and allow some generalization (Hitchcock et al., 2015).

As a full experimental design, single-case studies include the expected elements of control and randomization. In a multiple-baseline study, control is provided through the repeated measurements of a single subject comparing baseline data and intervention data (Barlow, Nock, & Hersen, 2009). Each subject in the experiment acts as her or his own control through comparison of the baseline to the intervention phase with all factors other than the independent variable being held constant (Kennedy, 2005; Kratochwill et al., 2013). Replication of results with respect to treatment effects is sought between subjects to increase external validity in the results of the study (Barlow et al., 2009).

In this study, subjects drawn from a single-case of in-service elementary teachers were compared with respect to response to different interface modalities used within computer science instruction. The interface modality of the instructional resource –
analog or digital – was the independent variable assessed. Instruments described in the following section were used to assess the subjects’ confidence about and competence with various aspects of computer science as encountered in a computational thinking instructional approach to the topic. The control for each subject was established during a baseline phase followed by a randomly timed intervention with additional measurements in a intervention phase. The start of the baseline phases were non-concurrent to accommodate the schedules of the subjects and the researcher. Given the highly personal nature of the study around professional development, however, the lack of concurrency was considered to have little impact.

Subjects were selected from multiple schools to reduce the threats to internal validity from teacher interactions. The study was designed to include six subjects though two subjects withdrew leaving an actual \( n \) of four. Subjects were randomly assigned to the two experimental groups, analog and digital, resulting in three subjects for each as planned and two each as implemented. One subject from each group withdrew after randomized assignment but before the actual study began. Within each group, subjects were randomly selected for the order in which they received the intervention. Subjects in the study underwent a total of eight assessments made up of three to five each for baseline and treatment phases depending on the randomized implementation timing. This randomization was recommended by Kratochwill and Levin (2010) to increase internal validity and to allow the use of additional statistical tests.

The intervention introduced teachers to basic concepts of computational thinking including commands given to robots, basic control flow statements, and functions. The role of functions as a method of bundling, naming, and then enacting a group of repeated
or commonly used commands was highlighted in the professional development. For the
digital group, the intervention used the Scratch Jr. application on iPad Mini tablets. The
analog group used the *Robot Turtles* board game. Both resources are designed for use by
primary grade students in kindergarten through second grade. Additionally, each tool had
a similar approach to defining and calling functions using a specific command; the
function frog card in *Robot Turtles* and the envelope/message block in Scratch Jr. The
professional development provided by the researcher covered the basic concept of a
function as a repeated pattern expressed as an algorithm as well as the specific method of
defining and calling a function in the resource being used.

**Research Context**

This study was conducted at rural elementary schools within the service area of
the Genesee Valley Educational Partnership in Western New York. The Partnership
serves 22 school districts across four counties with a total student population of 22,339.
There are 26 public elementary schools in the region with a total of 632.5 teachers and
11,521 students in kindergarten through sixth grade (GVEP). The districts in the region
tend to fall below average district wealth for New York State. The mean combined
wealth ratio for the region is 0.59 meaning that these districts are about 60% as wealthy
as the New York State average (nydatabases.com).

None of the elementary schools had established a formal computer science
program when this study began. Six of the schools had been involved with STEM
activities through a New York State Learning Technologies Grant run by the Genesee
Valley School Library System. As Director of the School Library System, the researcher
for this study was the principal investigator on the state grant as well, and so this study
avoided those six schools. The current state grant established the researcher as a credible and trusted provider of technology professional development in the region amongst principals and superintendents.

There are future opportunities for a computationally literate population in the Genesee Valley region. The region falls between the greater metro areas of Buffalo and Rochester. Both cities are investing heavily in scientific and technical industries including the integrated photonics hub in Rochester. Genesee County also has an emerging advanced manufacturing project that will provide hundreds of jobs for middle-skills workers who are computationally savvy (Spector & Sharp, 2015). To educate the workforce to meet future needs, the schools in the Genesee Valley region need to emphasize computational thinking as the foundation of computer science and other technology understandings. This study compared the efficacy of different approaches to professional development for in-service teachers to reach that goal.

**Research Participants**

The case being investigated for this proposed study was elementary teachers with limited computer science experience who were ideally teaching computational thinking for the first time. Given the younger age alignment for resources that were used, the sampling frame was limited to kindergarten and first grade teachers. The actual subjects were purposively selected from responses to a call for participation shared by principals from the region. The selection excluded those who had prior professional development or instructional experience in computer science or programming. This was intended to isolate the impact of the intervention and amplify any changes between the baseline and
the intervention measurements. Additionally, subjects were selected from a single gender to remove potential gender impacts.

**Instruments Used in Data Collection**

There were two primary instruments used within this study. Teacher confidence was measured using an adapted version of the Computer Programming Self-Efficacy Scale initially developed by Ramalingam and Wiedenbeck (1998) and modified by Kong (2017) for elementary student programmers. Teachers’ competence was measured using the Computational Thinking Test designed by Román-González (2015) that had undergone content and criterion validation (Román-González, 2015; Román-González et al., 2017). Both of these instruments were adapted for this study to better address the focus on teachers. Stewart, Thrasher, Goldberg, & Shea (2012), writing in the context of health research, noted that adaptation of an existing measurement is an acceptable practice in order to meet the specific self-reporting needs when investigating a smaller, distinct population.

**Computer Programming Self-Efficacy Scale for Confidence.** The Computer Programming Self-Efficacy Scale (CPSES) was initially developed by Ramalingam and Wiedenbeck (1998) to assess college students learning the C++ programming language. The CPSES has undergone numerous adaptations for use in studies of different populations and different programming languages. The adapted version used in this study was based upon modification by Kong (2017) for elementary student programmers. Additional adaptations by Kukul, Gökçearslan, and Günbatar (2017) for high school programmers and by Tsai, Wang, and Hsu (2018) for middle school programmers were also consulted. In this instance, minor to moderate context and content adaptations were
made according to the definitions used by Stewart et al. (2012) as compared to the CPSES adaptation validated by Kong. For example, the contextual focus of statements from Kong’s scale such as “I can code with. . .” were adapted to address an instructional context as “I can teach the use of. . .” (p. 99). Additional language adjustments were made to clarify terminology as introduced in the professional development for this study while maintaining the same underlying concepts of the CPSES version validated by Kong (2017).

The adapted Elementary Teacher Computer Programming Self-Efficacy Scale (see Appendix A) included 15-items that asked subjects to rate their level of confidence with statements about teaching computer programming. Each statement was rated on an 11-point scale ranging from 0 (Not at all Confident) to 10 (Highly Confident) based on the recommendations of Bandura (2006). Instructions were provided for the subjects asking them to rate their confidence about their ability to complete the instructional objective at the point in time of the measurement (Bandura, 2006).

**Computational Thinking Test for Competence.** The adapted Computational Thinking Test (see Appendix B) originally designed by Román-González (2015) included 28, multiple-choice questions covering a variety of computational thinking concepts. Given the strength of the validation studies for the current instrument, no questions were added or removed. However, permission was received (see Appendix C) to adapt the graphical representation of the questions to reflect the interfaces of both *Robot Turtles* and Scratch Jr. This moderate, content adaptation meant that teachers were assessed in the same graphical interface environment as they learned and taught (Stewart et al., 2012).
**Challenges for adapting the Computational Thinking Test.** Prior to the adaptations, additional consideration was given to the potential impact the incongruity of movement commands between the original graphical representation of the Computational Thinking Test and the adapted version using graphics from *Robot Turtles* and Scratch Jr. could have on the assessment. In the Scratch Jr. programming language, the kitten or other sprites are moved around the screen using absolute directional commands. This means that the command arrow ➡ means move to the right, not move forward. Even if the kitten is facing to the left, the ➡ command will move the sprite to the right one unit making it look like the kitten is moving backwards. This is different from other programming situations, such as the *Robot Turtles* board game, where movement commands are relative. In *Robot Turtles*, the forward movement command is always interpreted as movement of one unit in the direction that the turtle is facing. Therefore, movement is relative to the turtle’s point of view.

A search did not reveal any definite statement as to why the Scratch Jr. team decided to use absolute movement as opposed to relative movement commands as are found in the Scratch language. However, there are clear indications that this was a decision based on a desire to implement a coordinate grid system upon which movement would happen. “The grid was designed to help children understand the rules of measurement for each programming block. It addresses the countable unit of measurement for linear movement. For example, a character programed to ‘Move Right 10’ glides 10 grid cells rather than 10 pixels or an arbitrary unit” (Bers, 2017, pp. 122-123). The designers intended the movement to reflect movement students would see on a grid or number line where directions like up, down, left, and right would be used. This
interpretation is reinforced by the intentionality of having the script execute horizontally as opposed to vertically as would happen with text code. “This choice reinforces print-awareness and English literacy skills” (Bers, 2017, p. 119).

In terms of potential impact on this study, the issue was that the intended instrument for measuring competence used a mixture of absolute and relative movement commands. Of concern was that the absolute and relative questions were incongruous with the adapted graphics. Within the original instrument, the questions with absolute movement were graphically presented in a style more similar to the cards from *Robot Turtles* with arrows of movement while the relative movement commands were written using Scratch coding blocks. This was incongruous with what the subjects might expect from their experiences using the resources for the study with *Robot Turtles* using relative and Scratch Jr. using absolute movement. Research by Bruner and Postman (1949) on recognition in the case of incongruous stimuli suggested that the differences between absolute and relative movement commands should be small. While this study found a significant difference ($t = 3.76, p < .01$) in the time it took to recognize a playing card where colors of suits were reversed, the difficulty of a subject to identify an incongruous card dropped rapidly after repeated interactions with miscolored cards. This suggested that if subjects in this proposed study are informed before interacting with the measurement, and then are presented with example questions that introduced and reinforced the incongruous use of relative and absolute movement, the impact should be minimized.

A second potential impact was identified stemming from the incongruity of notation styles. The difference in symbol notation and representation of movement
between the proposed instrument and what the subjects in this proposed study learned in their respective interactions with Robot Turtles and Scratch Jr. was investigated. In writing about learning math, Bruner and Kenney (1965) noted that an understanding of the abstract, foundational concept was more important than the concrete representation of a mathematical interaction. Bruner and Kenney (1965) observed that the students “had not only understood the abstractions they had learned, but also had a store of concrete images that served to exemplify the abstractions. When they searched for a way to deal with new problems, the task was usually carried out not simply by abstract means but also by ‘matching up’ images” (Bruner & Kenney, 1965, pp. 56-57). This suggested that as long as the subjects in this proposed study had a chance to experience the multiple concrete images of movement commands as seen and explained in the example questions of the instrument, they should be able to match these images up with their abstracted understanding of different possible movement types. While this learning would need to be self-directed during the baseline assessment interactions, additional explanation could be provided during the intervention. During the professional development intervention, it was reinforced that the move type seen by the subject in the resource used – relative for those in the Robot Turtles group, and absolute for those in the Scratch Jr. group – was not the only type.

Based on this review, the potential impacts of the incongruity in iconographic representation and movement command style were judged to be of limited consequence to the study. As such, the adaptations were made to the graphical representations of the test questions and answer choices to use images from Robot Turtles and Scratch Jr.
Following the recommendations of Stewart et al. (2012), efforts were taken to maintain all other aspects of the questions whenever possible.

**Procedures**

The first step in the implementation of this study was to seek and obtain approval for the research from the St. John Fisher Institutional Review Board. Following approval of the experimental design, subjects were sought using a call for participants sent through principals as defined above. After selection, the researcher completed a randomization process as recommended by Kratochwill and Levin (2010) to determine placement within either the analog or digital group. A second randomization process was used to create an order of intervention for subjects within each group. Randomization was completed using the list randomizer from Random.org. The last step in the initial phase of the study was to meet with the subjects to inform them of the full parameters of the study and to seek consent. The multiple-baseline approach required additional explanation for subjects. During the consent meeting, subjects were reassured that during the baseline phase they may not be able to answer the questions on the instrument for assessing competence in computer science. It was explained that this was an expected part of the research design and that the subjects should not independently seek out information on the topic.

**Ethical considerations.** In terms of ethical considerations, subjects were notified that pseudonyms would be used for data collection, analysis, and reporting. They were given an opportunity to select a pseudonym at the time of informed consent. The pseudonym key was stored as an encrypted note in an industry standard password vault on the researcher’s phone protected with two-factor authentication until completion of the dissertation process and then was securely deleted. The fully anonymized data was stored
in an encrypted, non-synchronized, Dropbox folder in an account protected with two-factor authentication. It was also explained that the research findings would be disseminated in this dissertation and would be shared at conferences and in other possible publications. The findings were shared with the subjects at the conclusion of the research study.

**Study procedure.** This study was designed for implementation during an 8-day period. The 8 days for the study were scheduled non-currently to accommodate the calendars for the subjects and the researcher. In the initial meeting, following receipt of consent, the researcher presented the subjects with examples of the two instruments. At the start of the 8-day study period, a link to access the online instruments was sent to subjects via a daily email. Subjects completed a specified number of baseline assessments as indicated by the randomization process. On the 4th to 6th days of the intervention, again depending on randomized order, the researcher met with each subject to provide professional development on teaching with either the analog or digital resource. Two sessions ranging from 45 minutes to an hour took place on succeeding days. During and after the intervention, subjects continued to take the daily assessments. The subjects each completed a total of eight instruments throughout the study.

The first professional development (see Appendix D) focused on the use of directional commands within the analog or digital resource as a way to move the character around the play area. Basic conditional statements based on If/Then construction were also introduced. The instructional content and delivery was the same for both groups, only the resource presented to the subjects to illustrate the concepts being taught differed. The handout was used by the researcher as an outline to ensure
similarity in delivery. To address the potential issue of incongruency between the instruction and the assessment, both groups were shown examples of relative and absolute movement. During each professional development session, the subject was provided with potential lesson ideas that could be used to introduce the analog or digital resource in classroom instruction. Materials, either board games or iPads loaded with the Scratch Jr. software, were provided to subjects in numbers sufficient for whole class instruction.

After the first professional development, the researcher met again with the subjects on the following day for a second session of professional development on the use of advanced control flow and functions (see Appendix E). Again, the professional development was the same for both groups except for the illustrative use of the specific analog or digital resource. In the second session, additional focus was placed on explaining the role of computational thinking as a broad approach to computer science as a way to encourage confidence. For example, advanced concepts like functions and algorithms were introduced within the context of pseudocode as a way to describe programming using plain English to make the practice more relatable.

**Analysis**

It is generally accepted that baseline and intervention phases for single-case experimental design should include at least three measurements each for a total of at least six measurements for a multiple-baseline design (Kennedy, 2005, Kratochwill et al., 2013). Each subject’s performance can then be individually compared between the baseline and intervention phases to determine any treatment effect. Additional comparison between subjects provides additional evidence supporting the replicability
treatment effect. The instruments for this study were selected in part because their brevity made it possible to use them for the multiple assessments required to establish a stable baseline (Barlow et al., 2009). The requirement for multiple assessments made the use of more detailed instruments too onerous for the subjects. The exclusion of teachers with prior experience was intended to amplify the potential for change from the baseline to intervention. In this study, subjects were asked to complete a total of eight measurements on each of the two instruments with at least three baseline and three intervention measurements for every subject.

Traditionally, treatment effect in a single-case design has been assessed using visual analysis methods (Hedges, Pustejovsky, & Shadish, 2013). Graphs showing the data points from the baseline and treatment phases and are compared visually in terms of the level of difference, the immediacy of the change, the emergence of trends, and the fit of observed data to expected outcomes (Manolov & Moeyaert, 2017). This study made use of visual analysis of graphs to describe the subjects’ performance on the two assessments following an established procedure from Lane and Gast (2014). The procedure calls for comparison of the baseline and intervention phases using a variety of manually executed, visual processes such as first- and second-half median identification to reveal trends.

Recently, procedures for statistical analysis of single-case designs have been developed to allow more descriptive comparisons of effect sizes and to facilitate comparison of single-case results with other types of research (Hedges et al., 2013). Hedges et al. (2013) created a measure of effect within multiple-baseline studies that can be used in direct comparisons with Cohen’s $d$ or Hedges’ $g$. The DHPS package for SPSS
requires a minimum of three cases per study to complete analysis. With only two cases per group, this study used a similar package for R to calculate Hedges’ $g$ on an individual case basis. SSDforR is a recognized package of statistical analysis tools designed for use with single-case research published on the CRAN repository (Auerbach & Schudrich, 2013). SSDforR was used to identify Hedges’ $g$, a variation of Cohen’s $d$ indicated for small sample sizes, using both a standard and indexed method that accounts for trends in the baseline phase (Auerbach & Schudrich, 2013).

**Chapter Summary**

The use of a single-case experimental design for this quantitative dissertation allowed the research to be conducted using a smaller sample size while still maintaining experimental controls, randomization, and other standards of a full experimental design. Establishing a case of elementary teachers allowed potential generalization to similar populations as a way to provide recommendations for addressing a widespread need for professional development on computer science for elementary teachers. As an experimental design, there was also the potential for the study to reveal causation for either or both interventions. Results from the two assessments used in the study were explored using visual analysis as well as a version of Hedges’ $g$ customized for use with single-case experimental design. The results of this analysis are presented in the next chapter.
Chapter 4: Results

Research Questions

This study investigated two research questions developed from a review of existing literature that highlighted both a gap in understanding about elementary teacher professional development and the need for additional research around the potential impact of interface modalities. The two specific research questions were as follows.

1. Is the confidence of elementary teachers in an initial computer science professional development interaction different depending on the use of an analog or digital teaching tool?

2. Is the competence of elementary teachers in an initial computer science professional development interaction different depending on the use of an analog or digital teaching tool?

A qualitative study using a single-case experimental design methodology was developed to test those two questions. The single-case experimental design approach was selected as an accepted method for a randomized, controlled experiment with a smaller number of subjects (Hitchcock et al., 2015; Smith, 2012). The initial design called for six subjects, but after the start of the randomization process, two withdrew leaving a final count of four subjects. Subjects were randomly placed into two groups that received professional development on the same content using either an analog or a digital resource to accompany the instruction. The timing of the professional development was also randomized to happen after a 3- to 5-day baseline period to establish a control phase for
each subject as per accepted practices for the methodology (Kennedy, 2005; Kratochwill et al., 2013). The results of each subject were analyzed using both a visual analysis procedure widely accepted in the literature (Lane & Gast, 2014) and using additional statistical analysis methods specifically designed for use with single-case experimental design studies (Auerbach & Schudrich, 2013).

Visual Analysis

This study included four subjects, or cases, drawn from a larger case of kindergarten and first grade teachers from member districts in the Genesee Valley Educational Partnership. Though initially designed to include six cases, two subjects withdrew after the randomization process before the receipt of consent. One person from each group withdrew leaving an even number of subjects for each group. All of the subjects in this study were female. To increase internal validity, a restriction to a single gender was included in the selection criteria. Given the predominance of females within elementary school faculties, it was unsurprising that all potential subjects were female.

Ages and years of experience were not collected from subjects in part to remove potential discomfort associated with a male researcher asking this of a female subject and also because the literature suggested that age was not a factor for the research questions. Hsu et al. (2017) found in their study of games and technology in Singapore that results were not dependent on either age or gender. Similarly, Horn et al.’s (2009) observations of initial interface modality preferences for women and girls were similar regardless of age.

Each individual case is described below including adherence to the procedure, a description of the baseline and intervention phases, and a visual representation of the case data. Cases are listed using the pseudonym selected by the subject at the consent meeting.
For each case, a graphical representation of data is included. To facilitate visual analysis and a comparison between the two aspects of confidence and competence, both have been rendered as a percentage value. In the case of confidence, the percentage score was derived from the mean of the daily scores on the Elementary Teacher Computer Programming Self-Efficacy Scale multiplied by 10 to create a percentage value from the 0-10 scale. The competence value displayed indicates the percentage of correct items on the Computational Thinking Test for each day’s administration. The vertical line and break in series lines indicates the start of the intervention and separates the prior baseline phase results from the intervention phase results that follow the vertical line.

**Competence.** Not surprisingly, there was no real variation in the measurement of competence between the baseline and intervention phases. The intervention included two, brief, 45-minute, professional development sessions. In no case was there an observable change in the trend, level, or stability of the measurements of competence as per the established basic elements of visual analysis for single-case experimental designs (Kennedy, 2005). Across all cases, there was less than 10 percentage points of difference between the overall means of the competence scores (Range = 62.1 to 70.1). As seen in Table 4.1, all cases but one had higher scores but none of the differences were significant ($p < .05$). The professional development intervention for this study focused on the use of functions as a programmatic way to combine a set of multiple commands into a single, easily-implemented function command intended for repeated use within a program. Functions are directly implemented in bothScrath Jr. and Robot Turtles. Functions are one of the seven concepts addressed by questions in the Computational Thinking Test along with directions, repeat for loops, repeat until loops, if-then, if-then-else, and while
loops (M. Román-González, personal communication, August 5, 2017). The mean scores for baseline and intervention phases for each of the concept areas is presented in Appendix F. Table 4.1 shows that there was an increase in the mean competence score on the four questions addressing the use of functions, however the visual analysis of the graphs presented in Figure 4.1 also shows that the results are inconclusive. Despite increases in the mean scores, Figure 4.1 shows that there are no distinct differences between baseline and intervention phase scores. Specifically, the lack of significance in results is indicated by the high degree of variation within each phase and the high level of overlap in scores between phases (Kennedy, 2005).

Table 4.1

<table>
<thead>
<tr>
<th>Case</th>
<th>Baseline</th>
<th>Intervention</th>
<th>Baseline</th>
<th>Intervention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jumper</td>
<td>67.14</td>
<td>70.24</td>
<td>50</td>
<td>66.67</td>
</tr>
<tr>
<td>jlo</td>
<td>58.57</td>
<td>70.24</td>
<td>40</td>
<td>62.50</td>
</tr>
<tr>
<td>Taylor</td>
<td>70.54</td>
<td>68.75</td>
<td>31.25</td>
<td>68.75</td>
</tr>
<tr>
<td>Charlotte</td>
<td>58.93</td>
<td>65.18</td>
<td>18.75</td>
<td>31.25</td>
</tr>
</tbody>
</table>
Figure 4.1. Despite an increase in mean competence scores on questions addressing functions, visual analysis shows there was no clear baseline established and no clear indication of an intervention trend.

Not only was there no observable change in overall competence as shown for each case in Figure 4.2 through Figure 4.5, there was no observable impact on competence based on the interface modality for the professional development resource used with the subjects. The stability of the competence measurement does provide a stable point of comparison for observed changes in confidence in these figures, however. There were notable changes in levels of confidence both between the phases and between the interface modality groups.

The graphical results for confidence were analyzed for each case using a procedure defined by Lane and Gast (2014) including within condition or phase analysis and between condition analysis. The graphs for each case include split-middle trend
estimation lines for within condition analysis as indicated by Lane and Gast (2014). An additional table for each case describes other points of within condition analysis from Lane and Gast’s (2014) procedure including mean, median, range, and relative level change. The SSDforR package within R Studio was used to identify these statistics for each case. Following Lane and Gast’s (2014) procedure, the second half median of the baseline phase and the first half median of the intervention phase were compared to identify the level of relative change between conditions. The absolute change between the final measurement of the baseline phase and the first measurement of the intervention phase is also reported as per the procedure. None of the cases had any overlap of data between phases for the confidence measure, so this is not reported below.

**Digital – Jumper.** Jumper, like all of the cases in this study, is a teacher at a small, rural elementary school in the Genesee Valley Educational Partnership of Western New York. She teaches first grade and has no prior experience with computer programming. Jumper was randomly assigned to the digital group. She completed all eight assessments on succeeding days but failed to complete her third baseline assessment prior to the first professional development intervention as seen in Figure 4.1. As such, she ended up having two baseline measurements and six intervention phase measurements and so failed to meet established standards for creating a stable baseline across a minimum of three measurements in order to create a control comparison (Kennedy, 2005; Kratochwill et al., 2013).

With only two baseline measurements, it was impossible to follow the full procedure defined by Lane and Gast (2014) for within phase analysis. However, the split-middle method estimation for the intervention phase measurements of confidence
revealed a slight negative trend as shown by the dashed line in Figure 4.1 and the relative change of -1.9% shown in Table 4.1. Jumper was the only subject to demonstrate a negative trend for confidence in any phase. Without a valid baseline for comparison, it was also impossible to analyze many of the between phase aspects of the Lane and Gast (2014) procedure. One point that could be compared was the final measurement of the baseline for confidence (4.4%) with the initial intervention phase measurement (23.1%) showing an absolute level change of 18.7 percentage points.

![Figure 4.2. Percentage scores for measures of Confidence (Elementary Teacher Computer Programming Self-Efficacy Scale) and Competence (Computational Thinking Test) for Jumper during baseline and post-intervention phases including split-middle trend estimation lines.](image-url)
**Table 4.2**

*Within Condition Statistics for Jumper*

<table>
<thead>
<tr>
<th>Phase</th>
<th>$M$</th>
<th>Median</th>
<th>Range</th>
<th>Relative Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>2.2</td>
<td>2.2</td>
<td>0.0 – 4.4</td>
<td>--</td>
</tr>
<tr>
<td>Intervention</td>
<td>24.3</td>
<td>24.1</td>
<td>21.3 – 28.1</td>
<td>-1.9 Deteriorating</td>
</tr>
</tbody>
</table>

**Digital – jlo.** jlo (she used a lowercase j for most of the assessments) is a first-grade teacher who expressed some interest in learning more about computer programming based on a general interest in technology. She completed all eight assessments on the corresponding days as noted in Figure 4.2. jlo was randomly assigned to complete five baseline measurements and three intervention measurements. Though jlo began her baseline phase with the highest level of reported confidence, her measurements throughout the five-day baseline were stable with a relative change of +7.2 percentage points. As seen in Figure 4.2, the baseline trend as estimated by the split-middle method indicated less change than the trend for the intervention phase trend. Between condition analysis showed a relative change from the second-half median of the baseline phase to the first-half median of the intervention phase to have increased by 11.3 percentage points. Visual analysis of the between phase trends shows an increasing trend post-intervention as compared to during the baseline. Combined with an absence of overlapping data points between the phases and an overall increase in confidence measurement scores, the presence of an increasing trend suggested that jlo presents a strong case for further effect size analysis (Vannest & Ninci, 2015).
Figure 4.3. Percentage scores for measures of Confidence (Elementary Teacher Computer Programming Self-Efficacy Scale) and Competence (Computational Thinking Test) for jlo during baseline and post-intervention phases including split-middle trend estimation lines.

Table 4.3

*Within Condition Statistics for jlo*

<table>
<thead>
<tr>
<th>Phase</th>
<th>$M$</th>
<th>Median</th>
<th>Range</th>
<th>Relative Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>25.0</td>
<td>25.6</td>
<td>17.5 – 30.6</td>
<td>+7.2 Improving</td>
</tr>
<tr>
<td>Intervention</td>
<td>44.2</td>
<td>43.1</td>
<td>39.4 – 50.0</td>
<td>+10.6 Improving</td>
</tr>
</tbody>
</table>

**Analog – Taylor.** Taylor is a kindergarten teacher. Her husband works in a technology field, but she reported having no prior exposure to computer programming herself. Taylor completed all eight assessments on the correct days for her assignment to
a 4-day baseline phase and a 4-day intervention phase shown in Figure 4.3. She began the study with the second highest confidence score and had the greatest level of trending growth in confidence during the baseline phase as shown in Table 4.3. In terms of relative change seen in between phase comparison using the second-half median from the baseline and the first-half median from the intervention phase, Taylor showed an increase of 51.2 percentage points of confidence. There were no points of overlapping data and a strong absolute level of change of 26.4 percentage points from the final baseline measurement (38.6%) to the first intervention measurement (65.0%) making this case a possible point of evidence for identifying a functional relationship between the analog professional development modality and an increase in confidence (Vannest & Ninci, 2015). Byiers, Reichle, and Symons (2012) noted that even in the presence of a positive trend during the baseline phase, stability of the trend and differences between the expected trend and the results see after the intervention must also be considered. In the case of Taylor, the trend stabilized in the second-half of the baseline measurements after a single instance of increase between days two and three of the phase. Lane and Gast’s (2014) procedure calls for considering the stability of the second-half baseline trend as showing clear difference between the baseline and the continuation of growth throughout the intervention phase.

**Analog – Charlotte.** Charlotte is a first-grade teacher with no prior experience in computer programming. Charlotte had been assigned by the randomization process to complete five baseline assessments, but she failed to complete her fifth before the first professional development meeting. She ended up with four baseline measurements and four intervention measurements as shown in Figure 4.4. Charlotte exhibited the highest
Figure 4.4. Percentage scores for measures of Confidence (Elementary Teacher Computer Programming Self-Efficacy Scale) and Competence (Computational Thinking Test) for Taylor during baseline and post-intervention phases including split-middle trend estimation lines.

Table 4.4

*Within Condition Statistics for Taylor*

<table>
<thead>
<tr>
<th>Phase</th>
<th>M</th>
<th>Median</th>
<th>Range</th>
<th>Relative Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>28.4</td>
<td>29.1</td>
<td>16.9 – 38.8</td>
<td>+19.4 Improving</td>
</tr>
<tr>
<td>Intervention</td>
<td>81.9</td>
<td>85.6</td>
<td>65.0 – 91.3</td>
<td>+15.6 Improving</td>
</tr>
</tbody>
</table>
level of variability in terms of competence, but her baseline for confidence remained the steadiest of the four cases. Charlotte also demonstrated the most gains in confidence in the intervention phase. As seen in Table 4.4, Charlotte had the highest level of relative change within the treatment phase (+20.9 percentage points). Table 4.5 shows that Charlotte also had the highest level of relative change between phases as measured by comparing the second-half median of the baseline and the first-half median of the intervention phase (Lane & Gast, 2014). Charlotte presented an ideal data set for single-case visual analysis with a stable baseline lacking any trend followed by an immediate and pronounced difference and positive trend following intervention (Vannest & Ninci, 2015).

Figure 4.5. Percentage scores for measures of Confidence (Elementary Teacher Computer Programming Self-Efficacy Scale) and Competence (Computational Thinking Test) for Charlotte during baseline and post-intervention phases including split-middle trend estimation lines.
Table 4.5

*Within Condition Statistics for Charlotte*

<table>
<thead>
<tr>
<th>Phase</th>
<th>$M$</th>
<th>Median</th>
<th>Range</th>
<th>Relative Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>15.5</td>
<td>15.3</td>
<td>12.5 – 18.8</td>
<td>-0.31 Deteriorating</td>
</tr>
<tr>
<td>Intervention</td>
<td>69.5</td>
<td>76.3</td>
<td>43.8 – 81.9</td>
<td>+20.9 Improving</td>
</tr>
</tbody>
</table>

**Between case visual analysis.** The procedure for visual analysis specified by Lane and Gast (2014) also establishes a method for comparing baseline and intervention phase conditions across cases. To understand the relative change between phases for each case, Lane and Gast (2014) call for a comparison of the second-half baseline median with the first-half intervention median. This attempts to control for any trend in the baseline by looking at a median of the later-half of measurements as compared to the median of the initial-half of intervention measurements. The relative change between phases identified using this method are presented in Table 4.5. As can be seen, while all cases showed improvement as indicated by a positive difference between the baseline and intervention, the improvement was roughly twice as pronounced in the analog cases for Taylor (+35.9) and Charlotte (+43.8) as compared to the largest improvement in the digital group (+20.6). The difference can be clearly seen in the graphs, but this additional analysis from the Lane and Gast (2014) procedure clarifies the actual level of difference in the change. Further insight into the actual differences in the changes within and between cases were also revealed through statistical analysis.
Table 4.6

*Between Condition Statistics for all Cases*

<table>
<thead>
<tr>
<th>Case</th>
<th>Second-half</th>
<th>First-half</th>
<th>Relative Change Between Phases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline Median</td>
<td>Intervention Median</td>
<td></td>
</tr>
<tr>
<td>Jumper</td>
<td>4.4</td>
<td>25</td>
<td>+11.25 Improving</td>
</tr>
<tr>
<td>jlo</td>
<td>28.1</td>
<td>39.4</td>
<td>+20.6 Improving</td>
</tr>
<tr>
<td>Taylor</td>
<td>38.1</td>
<td>74.1</td>
<td>+35.9 Improving</td>
</tr>
<tr>
<td>Charlotte</td>
<td>15.3</td>
<td>59.1</td>
<td>+43.8 Improving</td>
</tr>
</tbody>
</table>

**Statistical Analysis**

Following the traditional visual analysis of single-case experimental design results as described in a procedure from Lane and Gast (2014), recent best practice has also suggested the use of an effect size analysis to allow for comparison of single-case results with other between group studies (Vannest & Ninci, 2014). One example of an analytical method that makes use of Cohen’s $d$ and Hedges’ $g$ comes from Shadish, Hedges, and Pustejovsky (2014). Hedges et al. (2013) designed a single-case optimized macro package for SPSS to determine an effect size using Cohen’s $d$ and Hedges’ $g$ as indicated by the sample size. That macro package requires a minimum of three cases per group to be analyzed and so was not able to be applied for this research. Auerbach and Schudrich (2013) offer a similar package for R Studio that can also compute effect size for Cohen’s $d$ and Hedges’ $g$ comparing phases between a single case.
The SSDforR package (Auerbach & Schudrich, 2013) available from the CRAN repository was used to compute effect size for the cases within this study as presented in Table 4.6. Hedges’ g was used to determine effect size based on the g index version of the procedure in SSDforR to account for possible trends in the baseline phase as indicated by Auerbach and Schudrich (2013). However, the effect size for Jumper had be disregarded due to a failure to establish an acceptable baseline for comparison. The next step was to ensure that the calculated effect sizes could be considered statistically significant using the two-sample t-test procedure within SSDforR (Auerbach & Schudrich, 2013).

Subjects from both the digital and analog groups showed statistically significant growth verifying the expectation from visual analysis. From the digital group, jlo showed a statistically significant difference between the mean of baseline confidence measures ($M = 25.0$) and the mean of intervention measures ($M = 44.1$), $t(6) = 5.24, p < .05$. There was a medium effect size as measured by the indexed Hedges’ g ($g = .4$) controlling for baseline trend, but a large effect size based on a traditional Hedges’ g ($g = 3.33$). In the analog group, Taylor also showed a statistically significant difference between baseline ($M = 28.4$) and intervention confidence measures ($M = 81.9$), $t(6) = 6.56, p < .001$, with a medium effect size ($g = .5$) for the indexed test an a large effect size on the regular test for g ($g = 4.03$). Charlotte from the analog group showed the most marked change. Charlotte showed a statistically significant difference between baseline ($M = 15.5$) and intervention confidence measures ($M = 69.5$), $t(6) = 6.12, p < .001$, with a large effect size for both the indexed ($g = .75$) and regular tests ($g = 3.76$). Both subjects from the
analog group were found to have experienced a larger change with a greater effect size than the digital group confirming the results identified during visual analysis.

Table 4.7

*Two-Sample T-Test and Effect Size Calculations*

<table>
<thead>
<tr>
<th>Case</th>
<th>Baseline M</th>
<th>Intervention M</th>
<th>t(6)</th>
<th>p</th>
<th>95% CI LL</th>
<th>95% CI UL</th>
<th>Hedges’ g index regular</th>
</tr>
</thead>
<tbody>
<tr>
<td>jlo</td>
<td>25.0</td>
<td>44.1</td>
<td>5.24</td>
<td>&lt; .05</td>
<td>-28.11</td>
<td>-10.22</td>
<td>0.4 3.33</td>
</tr>
<tr>
<td>Taylor</td>
<td>28.4</td>
<td>81.9</td>
<td>6.56</td>
<td>&lt;.001</td>
<td>-73.37</td>
<td>-33.51</td>
<td>0.5 4.03</td>
</tr>
<tr>
<td>Charlotte</td>
<td>15.5</td>
<td>69.5</td>
<td>6.12</td>
<td>&lt;.001</td>
<td>-75.68</td>
<td>-32.45</td>
<td>0.75 3.76</td>
</tr>
</tbody>
</table>

**Summary of Analysis**

The analysis of findings in this study was begun using traditional visual methods following a procedure developed by Lane and Gast (2014). Each case was considered individually to assess the level, trend, and stability of measurements during the baseline and intervention phases (Kennedy, 2005). For a more nuanced understanding of the results, additional statistical analysis was performed using the SSDforR package within R Studio (Auerbach & Schudrich, 2013). Two-sample t-tests were performed to identify statistical significance. Tests of Hedges’ g indicated effect size for each case as well. Excepting the case of Jumper who failed to establish a baseline to allow statistical analysis between phases, all other subjects were found to have statistically significant growth post intervention with medium to large effect sizes.

Further visual analysis between the digital and analog groups revealed additional insights. Perhaps most notably, for the two subjects in the analog group, intervention
phase measurements of confidence surpassed the level of competence. On the other hand, neither subject in the digital modality group ever achieved parity between measurements of confidence and competence. In prior studies, Curzon et al. (2014) reported an increase in confidence amongst teachers who attended a workshop on computational thinking, but in this study, there were marked differences in the level of confidence gained depending on the interface modality used. Geist (2015) found that a teacher’s level of confidence about math had a direct relationship on instructional practices. In this study, despite all subjects having similar levels of competence as measured by the Computational Thinking Test, only the two subjects in the analog group managed to close the gap between their demonstrated competence and perceived confidence. The implications of will be explored in the final chapter.
Chapter 5: Discussion

Introduction

This research used a single-case experimental design study to investigate potential differences in impact on confidence and competence based on the use of either an analog or digital instructional resource during professional development for elementary teachers on computer programming. There is a mounting pressure for teaching computer science and programming in elementary grades, yet elementary teachers are often ill-prepared to meet this challenge. A lack of qualified teachers was reported by Gallup and Google (2016) as the primary reason for not offering computer science instruction in elementary schools. The issue of teacher preparation is becoming more pronounced as additional states adopt new computer science standards that include elementary grade instruction. A study by Duncan et al. (2017) on the implementation of elementary computer science instruction in New Zealand explained the concern: “The teachers we have worked with throughout this, and previous studies, were more often than not anxious about teaching CS and programming concepts, and think they are not capable of doing this well” (p. 5). The question of how in-service teachers can best develop confidence and competence as teachers of computer science is, therefore, both timely and of important to the field.

This study investigated how the use of different types of instructional resources might impact a teacher’s development of confidence and competence as a leader of computer science instruction. Following identified best practice for computer science instruction in elementary grades, the professional development focused on the underlying
aspects of computer science referred to as computational thinking as described by Wing (2006) as opposed to a specific programming language (Fletcher & Lu, 2009; Grover & Pea, 2013). The decision to compare analog and digital resources was based on prior research showing an initial preference by both children and adult females for tangible or analog interfaces over digital (Horn et al., 2009). Additional research showed that using analog resources could increase both confidence and competence (Schneider et al., 2011). This study also drew upon research on the role of playfulness and games in developing confidence and competence in general (Phillips et al., 2014) and in the specific case of computational thinking (Berland & Lee, 2011).

Four kindergarten and first grade teachers from the small, rural districts of the Genesee Valley Educational Partnership in Western New York were selected as subjects to be evaluated within the case of elementary teachers. The subjects were randomly assigned to either an analog or digital group, and then randomly assigned to an order for intervention after the establishment of a baseline phase. The multiple measurements taken within the baseline phase function as the control for each subject within single-case experimental studies. This inclusion of control and randomization allows the method to be considered a full experimental design with results that can be generalized and with the potential for establishing a functional relationship between the intervention and outcomes. The results of this study were analyzed using the traditional visual methods called for by single-case experimental design as established in a procedure by Lane and Gast (2014). Additional statistical analysis was also performed to establish statistical significance and identify an effect size to allow comparisons with other studies using a variation of Hedges’ g optimized for single-case design (Auerbach & Schudrich, 2013).
The professional development intervention consisted of meetings with the teachers for about 45 minutes on succeeding days after a period of baseline measurements. The content of the professional development was the same for each group, only the instructional resource used to illustrate what was being taught and discussed differed. For the analog group, the instruction used the *Robot Turtles* board game. The digital group used the Scratch Jr. iPad app. The teachers were provided enough of either resource to use them for whole-class instruction during the intervention phase of the study if desired.

Unsurprisingly, given the brief intervention, there were no observable results on the teachers’ competence as measured by the Computational Thinking Test developed by Román-González (2015). There were observable differences in terms of confidence. The results from this study showed that in every case the teachers experienced a significant growth in their confidence to teach computer science after the intervention as measured by the Elementary Teacher Computer Programing Self-Efficacy Scale adapted from a scale originally developed by Ramalingam and Wiedenbeck (1998). Notably, the changes in confidence were more pronounced for the analog group as compared to the digital group. These findings corroborated previous findings as explored in Chapter 2 around interface modalities (Schneider et al., 2011), the use of games and play for professional development (Phillips et al., 2014), and the importance of initial exposure to computer science (Duncan et al., 2017). The higher level of positive change in terms of teacher confidence for the analog group suggests some changes to our current practices around professional development and instruction on computer science in elementary schools.

**Implications of Findings**
The strength of the effect size for the analog interface modality, combined with replication across two cases with visually similar responses to intervention with no lag, and the high level of change in reported confidence provide initial evidence for the existence of a functional relationship (Vannest & Ninci, 2015). Additional replications are required for an explicit declaration of causality; however, this study suggested a relationship between the use of an analog instructional resource during professional development and increases in teacher confidence about teaching computer science and computational thinking. At the least, this study confirmed findings from prior research around interface modalities and the use of games and play for learning. The findings from this study also provided additional support for and understanding of the theory of cognitive acceleration as an instructional approach (Adey & Shayer, 1994).

**The hidden importance of competence.** Of particular note in terms of implications of the findings was the visual evidence for the efficacy of the analog-based professional development for closing a gap between subjects’ demonstrated competence and perceived confidence. The lack of impact on competence was not unexpected given the brevity of the intervention, yet the act of measuring competence may have played an important role in this study. During the professional development sessions, the subjects seemed to be almost solely focused on the competence assessment based on their lack of discussion about the confidence assessment. Instead, subjects wanted to talk about their performance on the competence measurement, perhaps because it more closely mirrored their expectation of an assessment from their experience as teachers. Bandura (2006) warned that self-efficacy measurements can be impacted by a subject’s concerns over being judged. If the subjects in this study were indeed more focused on the competence
instrument, it may have helped alleviate some concerns about the confidence assessment. The implication is that measuring competence concurrently with competence is likely beneficial, even though no change in competence is expected.

In addition to the potential benefit of reducing concerns about the confidence measurement, the competence assessment also establishes a clear target for confidence growth. The two subjects in the analog group for this study were seen to close the confidence gap between demonstrated competence and perceived competence as shown in Figure 5.1. This is important given prior research on the impact of confidence on instructional practices in both computer science and the related field of mathematics. Geist (2015) found that math teachers who lacked confidence in their own mathematics abilities taught math less. Therefore, identifying professional development approaches that can increase teacher confidence in computer science to at least match their demonstrated levels of competence will potentially help increase instruction. This study showed that using an analog board game during computational thinking professional development resulted in the greatest growth of confidence and the only cases where the confidence level reached and surpassed the demonstrated competence. Both the interface modality and the content of the professional development intervention need to be more fully explored in terms of existing literature to reveal the nuances of this implication.
Figure 5.1 Subjects within the analog group showed a closing of the confidence gap between demonstrated competence and perceived competence as indicated by confidence. Confidence surpassed competence in the second post-intervention measurement for both.

**Computational thinking approach.** Prior research and recommended best practice for teacher computer science at the elementary level focused on computational thinking (Fletcher & Lu, 2009; Grover & Pea, 2013). This study corroborated those findings as well as the core definition of computational thinking as advanced by Wing (2006) as a fundamental skill accessible to all. In this study, all of the subjects demonstrated a reasonable level of mastery of computational thinking skills based on mean scores across measurements (range = 62.1 to 70.1). Even in the first measurement, prior to any professional development, subjects showed similar mastery ($M = 67.0$). The comparison of mean scores across the seven content areas of computational thinking as measured by the Computational Thinking Test (Appendix F) indicate that some skills and content may be more intuitively understood. Subjects scored higher on the directions, repeat loops, if-then, and if-then-else questions and struggled more on until loop, while loop, and function questions. The implication is that additional professional development may be needed on the second set of skills identified here. The first set seems to be more intuitive and so may be able to receive less attention during professional development sessions. Additional research is needed, however, to clarify these initial findings as such analysis fell outside the scope of this study.

Another related implication for teaching computational thinking is that highlighting the intuitive nature of some skills may be beneficial for teacher confidence.
As Wing (2006) pointed out in her seminal article, a critical aspect of computational thinking is that it is “a way humans solve problems” (p. 35). During the professional development sessions, this aspect was also highlighted. The subjects were reminded that humans designed computer hardware and programming languages and therefore it is unsurprising that computers function in a way that mirrors fundamental human thought.

The increased impact on teacher confidence seen in the teachers of the analog group that used Robot Turtles may be partially explained by the more fundamental link between the board game and computational thinking. Fletcher and Lu (2009) suggested that early instruction in computational thinking should focus on “development of human computing skills” as opposed to any “particular programming language” (p. 24). Despite being very easy to understand and use, Scratch Jr. is still a programming language. Instead, Fletcher and Lu (2009) stated, computational thinking should be presented through “vocabularies and symbols that can be used to annotate and describe computation” (p. 24). As an informal symbology for computation, the cards of Robot Turtles meet this definition. As an analog resource, the cards do not have any inherent procedural implementation within the game. Instead, the user must interpret the meaning and implement the action represented by the card. In contrast, the symbols of Scratch Jr. are more formal iconography for pre-defined commands within a structured language that is automatically implemented within the restraints of a digital application. As a result, the implications for these resources on computational thinking instruction are intertwined with the interface modalities each represented.

**Understandings of interface modalities.** What remains to be tested is whether this capacity for more direct computational interaction is inherent to the analog interface
modality or is an aspect of the symbiological design of the *Robot Turtles* game. However, the results from this study, when considered in the context of prior research on interface modalities, suggest that the functional relationship may be inherent to the interface modality in general as opposed to a specific resource. Of note, the results from this study corroborated prior research by Horn et al. (2009) showing a preference for an analog interface among women. Teachers in this study that were provided the same professional development using an analog resource showed roughly twice the positive change in confidence as compared to those using a digital resource. The research from Horn et al. (2009) suggested that the difference in growth of confidence might be due in part to an initial comfort the teachers could have felt when confronted with an analog view of computer science concepts. The implications around teacher comfort with the analog resource also corroborated the research of Schneider et al. (2011) who studied the differences between analog and digital interfaces in a work environment. In that study, Schneider et al. (2011) found that the analog interface resulted not only in more effective work, but also in an increased attitude of playful interaction as a measurement of the subjects being in a state of flow as defined by a confident, enjoyable, and engaged interaction with the work.

The results from the analog group of teachers in this study similarly suggest that those teachers entered into a state of confident, enjoyable, and engaged learning about computational thinking. The tangibility of the resource, enhanced by an inherent preference for an analog interface, and a state of playful flow resulting from the use of the board game seem to combine into an ideal professional development setting. The unlocking of the teachers’ confidence was seen to be immediate and significant in the
case of the subjects presented with *Robot Turtles*. This has implications for our current professional development and instructional practices that will be further explored in the recommendations section that follows.

**Playful approaches to computer science.** The aspect of playfulness was identified by Schneider et al. (2011) as an indication of a subject entering into a state of flow. Flow is a concept related to the idea of the zone of proximal development as included in the theory of cognitive acceleration defined by Adey and Shayer (1994). Flow and the zone of proximal development have also been combined into a single concept, the zones of proximal flow, as developed within the field of computational thinking by Basawapatna, Repenning, Koh, and Nickerson (2013). This study presented subjects with professional development interactions that were designed around a playful approach to computer science. Building from prior research by Cetin (2016) that showed greater gains for pre-service teachers that learned programming through playful interactions with Scratch as opposed to traditional instructional methods, this study extended the literature through a comparison of different aspects of play. Scratch Jr. is a playful environment, but *Robot Turtles*, being a game, adds additional scaffolding elements to the play space.

Play, like flow, is a state of mind wherein the player is feeling pleasure and engagement (Cailliois, 1958/2001). Games extend the state of play to add an additional aspect of challenge. Through a scaffold of rules, goals, and a feedback system, games structure the play to support successful navigation of a specific problem scenario (Suits, 2005). In terms of this study, the game aspect of *Robot Turtles* may have provided an additional boost to teacher confidence. As a game, *Robot Turtles* provided a framework for instruction that the teacher could follow as opposed to the more open-ended play of
Scratch Jr. This corroborates prior research on teacher confidence with games as an instructional approach (Hsu et al., 2017). Hsu et al. (2017) also caution, however, that teachers need instruction in the pedagogical aspects of games and game play in order to be most effective. This suggests that time should be given in professional development not only to the computational thinking concepts addressed within Robot Turtles but also to the game itself.

**Cognitive acceleration and computer science.** The theory of cognitive acceleration is based on the synthesis of ideas from Piaget and Vygotsky around how we acquire new learning (Adey & Shayer, 1994). This study seems to corroborate the inclusion of Piaget’s insistence on the need for a concrete approach when developing a schema for understanding new knowledge (Adey & Shayer, 1994). The analog group experienced professional development using a more concrete resource – both in terms of tangibility and direct relationship between content and interaction – than the digital group. The direct symbolic relationship and tangible interaction without an intermediate layer of digital hardware may also have implications on a learner reaching the zone of proximal development required for effective implementation of cognitive acceleration (Shayer & Adey, 2002). In the case of Scratch Jr., subjects had the extra layers of the tablet hardware and use of the digital touch commands of the app as additional barriers to entering the zone of proximal development. The need for game knowledge identified by Hsu et al. (2017) addressed digital games. In the digital environment, additional attention must be given to teaching the digital technology. When using analog resources, there is a more concrete connection to the content and likely a lower threshold for learning about the resource itself allowing more time to be devoted to the content. The additional
implication of this is that analog resources may therefore be less of a distractor for the social discourse that is so prominent in cognitive acceleration.

This study suggests that board games may be a more effective resource for facilitating the social discourse required in the cognitive acceleration phases of introducing cognitive conflict and applying social construction to resolve the conflict (Shayer & Adey, 2002). Unlike the individual interaction inherent to an app on a tablet, board games encourage the development of dynamic social interactions amongst the group of players. The social interaction is especially pronounced during cooperative board games (Berland & Lee, 2011; Peppler et al., 2013), but is present even in competitive games due to the interface modality. While digital interfaces build a two-factor visual interaction between the player and the screen, board games encourage a three-factor mode of interaction that includes the player, the other players, and the game board. The use of analog resources like board games would therefore likely be more effective for a cognitive acceleration approach.

A final implication from this study in terms of connections to the theory of cognitive acceleration concerns the role of the teacher as the expert facilitator of the cognitive acceleration process. Adey (2008) imbues the teacher with a high level of autonomy to pace classroom instruction according to the abilities of the students. The example cognitive acceleration lessons provided by Adey (2008) for math and science and by Hamaker and Backwell (2003) for technology are more complex than traditional instructional plans. Hamaker and Backwell (2003) stressed that “teaching for cognitive acceleration is risky” because the teacher must fill a complex role as “director of the activities, of the classroom dynamic and of the resultant discussion that follows” (p. 4). A
critical aspect of board games as opposed to digital games is the ability of the teacher to intentionally modify the board game experience to directly align with instructional objectives. In terms of this study, accommodations had to be made for demonstrating if-then conditional statements as neither resource includes them. It was easier to modify Robot Turtles to describe an if-then situation as the change simply required additional description of modifications to the cards, and rules of play. As a digital app, Scratch Jr. was impossible to modify. The intention of an if-then statement could be described, but not effectively demonstrated. In terms of directing learning within the cognitive acceleration approach the analog board game was more effective within this study.

**Additions to the literature.** A comparison of the ability to modify analog and digital resources as an aspect of facilitated instruction was one significant addition from this study to the literature around both cognitive acceleration and the instructional use of games. The malleability of board games in terms of meeting intentional instructional objectives has been discussed before in anecdotal terms (Harris & Harris, 2015) but this study presents more empirical evidence around the efficacy of board games for instruction. This aspect is but one part of the larger addition to the literature around professional development on computer science.

This study presents some of the first empirical evidence comparing the efficacy of approaches to computer science professional development for in-service elementary teachers. Menekse’s (2015) meta-analysis of studies on teacher professional development about computer science found none that directly addressed elementary teachers. This study addressed that gap by investigating the unique nature of computer science instruction in elementary schools. The results presented here show that teachers likely
possess a higher level of computer science competence than they believe they possess. This is especially true in terms of computational thinking concepts such as giving directional commands to robots and resolving if-then or if-then-else conditional statements.

The comparison of effects based on the same professional development content being delivered using either a digital or analog resource are also new additions to the literature. Prior studies such as those by Horn et al. (2009) and Schneider et al. (2011) found analog or tangible interfaces to be preferable and more effective in other settings. This study corroborates those findings within the context of in-service teacher professional development around computational thinking and computer science. Furthermore, the use of a single-case experimental design allows these findings to be generalized and to serve as the basis for establishing a functional relationship through replication. This study also provides evidence for an instructional approach to computational thinking that does not involve coding and computers, a lack identified by Yaşar (2018). The experimental nature of these findings, including elements of randomization and control, also gives them a strong foundation for use as the basis of the recommendations that follow.

**Recommendations and Future Research**

This study provides a foundation for the identification of a functional relationship between the use of an analog resource during professional development and greater gains in terms of teacher confidence around computational thinking. Additional replications are required, however, to firmly establish causality. Future research comparing the impact of analog and digital resources on teacher confidence and competence are therefore
necessary. Single-case experimental design research that establishes a pattern of similar results across even a few cases can still be used to inform practice based on emerging evidence of an observable functional relationship (Vannest & Ninci, 2015). In this study, the cases provided both visual and statistical evidence for the efficacy of analog resources for building teacher confidence during professional development about computer science. Recommendations for professional development, classroom instruction, and broader policy are presented in this section based on that evidence for the greater efficacy of analog board games. Recommendations for future research are also presented here.

**Future research ideas.** The results from single-case experimental designs are strengthened through replications and comparisons of similar results across cases (Kennedy, 2005). While implications and recommendations can be founded on this single study showing clear results across four cases, additional replications will strengthen the findings. Specifically, replications would answer lingering questions inherent to any such research. Do the different effect sizes found in this study hold true in similar situations? Do the different effect sizes hold true in the comparison of other analog and digital resources beyond Robot Turtles and Scratch Jr.? Additionally, this study looked at a single gender, but future studies might explore whether there are differences based on gender or age.

Another recommendation for future research concerns the specific skills and concepts within computational thinking. Comparison of results from the Computational Thinking Test in this study disaggregated by concept (see Appendix F) showed that some concepts may be more intuitive while others may require more explicit instruction. Future researchers should explore this further as the identification of intuitive concepts could
both provide early and easy successes to build teacher confidence and inform the development of a scope and sequence for teacher learning. Additional research is also needed to verify the validity of the Computational Thinking Test as an instrument for in-service elementary teachers. The initial validity testing for the instrument was completed with middle school students. In this study, the subjects did not appear to reach a measurement ceiling for the test as a whole suggesting validity, but additional studies could confirm this. Validity testing of the adapted Elementary Teacher Computer Programming Self-Efficacy Scale is also needed.

Finally, the results from this study regarding the efficacy of a board game for increasing teacher confidence in a technical subject provide another path of potential research. As noted earlier, further research is needed to clarify whether the analog nature of Robot Turtles, the use of game elements to structure play while using Robot Turtles, or some combination were most impactful on teacher confidence. In other words, do analog resources that are not games have the same impact on confidence as Robot Turtles? Or do games that are not analog have a similar impact on confidence? Research along this line would inform the selection and development of resources for professional development.

**Recommendations for professional development.** Given the results seen in this research, it is recommended that professional development for elementary teachers on computer science should use analog board games as an introductory resource. The evidence from this study corroborates the initial preference by women for analog interfaces to computer science and robotics as identified by Horn et al. (2009). As such, beginning a teacher’s professional development experience around computer science with an analog board game will be more likely to establish a mindset conducive to learning.
This study shows that for at least some concepts presented in computational thinking as a foundation for computer science, teachers are already quite competent. Using a board game as an instructional resource can help teachers close the confidence gap seen in this study. After initial successes and an increase in confidence, teachers will likely be more receptive for further professional development in additional aspects of computer science.

Equally important is the approach to introducing computer science in concrete terms as indicated by cognitive acceleration (Shayer & Adey, 2002). Concrete here refers to an explicit and direct connection between the idea and the illustration of the idea within the resource. The more direct depiction of computer science in an analog resource like Robot Turtles that is unencumbered by the additional layer of abstract complexity introduced by the computer hardware inherent to a digital resource results in analog resources being more concrete. The use of computational thinking as a foundational approach to computer science concepts is also important both for professional development and classroom instruction. The skills and concepts introduced in the professional development for this study are transferable to other subjects as well as whatever programming language a teacher or student encounters. The professional development for this study also modeled how skills like giving directions or resolving if-then conditional statements could be expressed in a tactile-kinesthetic through body movements. Having students resolve programming instructions through movement responses is also a way of introducing the concept of stepping or tracing through code as identified by Lister (2011) as a gateway for moving from concrete to formal operational thinking within computer science.
The final recommendation for professional development, at least in the earliest introduction of computer science topics, is to focus on similarities with other subjects. Activating prior knowledge and showing connections to other schema are both indicated by cognitive acceleration as ways to both increase learning and transfer (Shayer & Adey, 2002). The professional development used for this study (see Appendix D and Appendix E) highlighted how computer science is related to how humans intuitively think through problems. The primary teacher subjects identified with the need to make explicit if-then statements and establish the definitions of functions within their classrooms. They also seemed to respond positively to the explanation of if-then conditional statements as being similar to the cause and effect concept within reading instruction. Where possible, teacher professional development around computational thinking should be embedded into content area instruction as a process of critical and systems thinking. This integrated approach to computational thinking also introduces additional recommendations for classroom instruction.

**Recommendations for classroom instruction.** Increased attention is being given to teaching coding and programming in elementary schools, but in many cases the instruction fails to address the foundational skills of computer science that would allow students to truly understand the code they are writing (Fletcher & Lu, 2009). This code-first approach to teaching computer science could be compared to a phonics-based approach to teaching reading. While a student who masters all of the rules of phonics might be able to correctly call out every word on the page of a book and even pronounce nonsense words on an assessment of reading ability, she or he may in fact have no comprehension of the words being spoken. Approaching computer science instruction
through the building blocks of computational thinking as opposed to code provides both students and teachers an opportunity to gain comprehension. The recommendation to begin with computational thinking is not new. Fletcher and Lu (2009) opined that the code-based approach to computer science was detrimental to long-term success. More recently, Yaşar (2018) called for a separation of computational thinking from computational practices on electronic devices. This study presents a method for accomplishing this goal, a gap in the knowledge identified by Yaşar (2018).

Based on the findings from this study considered alongside prior research, it is therefore recommended that computer science instruction in elementary schools begin with a focus on computational thinking. Instruction should be based around analog resources such as Robot Turtles and including the students’ bodies as they engage in tactile-kinesthetic programming activities as suggested by Unnikrishnan (2016). Furthermore, the approach to teaching computational thinking should be interdisciplinary and integrated into other subjects of classroom instruction as a way to highlight the critical and systems thinking inherent to computational thinking. Ideally, the instruction will follow the framework established in cognitive acceleration (Shayer & Adey, 2002) where new learning is built around resolving cognitive disruption through social discourse. Analog board games, especially those with a cooperative approach as can be applied to Robot Turtles, are especially well suited to discourse and student engagement (Peppler et al., 2013). Also supporting the use of analog resources are the findings from Horn et al. (2009) that showed an initial preference for analog interactions to a robotics exhibit at a museum was at present for boys and girls as well as adult
women. Digital resources have a place in later instruction, but at least the initial introduction of computational thinking should be based around analog resources.

Horn et al. (2012) concluded that all types of interfaces serve a purpose in teaching computer science; graphical interfaces were especially useful for individual, self-guided, student work, but analog or hybrid interfaces were best suited for teacher-led instruction. In many elementary classrooms, however, Code.org or a similar web-based coding curriculum is likely the first or only instructional resource. Following the findings from Horn et al. (2012) it is instead recommended that instruction by the teacher in elementary classrooms should make use of analog resources or tactile-kinesthetic activities to introduce computational thinking concepts. This is especially important in terms of applying cognitive acceleration where the role of teacher as facilitator is best met using a resource that can be modified as needed to meet instructional objectives. In later, independent practice for students, digital resources like Scratch Jr. and Code.org can provide pathways for personalized learning, but they are not recommended as the sole instructional resource or for use as the introduction to new computational thinking concepts.

Recommendations for policy and standards development. In order to achieve the vision proposed by Wing (2006) of computational thinking being included alongside reading, writing, and arithmetic in elementary classrooms, policy makers need to understand what computational thinking is and how it fits within computer science instruction. Similarly, for the above recommendations on professional development and classroom instruction to be widely adopted, awareness of analog instructional resources and practices must be raised. Yaşar (2018) called for a change in computational thinking
instruction yet stated that a “decade of discourse and experimentation has yet to produce ways to separate CT from programming and the use of electronic devices” (p. 33). This study establishes evidence for the efficacy of computational thinking resources that do not involve electronic devices or programming within a specific language. Additional evidence for this method is available from countries like New Zealand that have taken an unplugged approach to teaching computer science for a number of years (Duncan & Bell, 2015). It is therefore recommended that policy and standards developing bodies look take an analog-first approach when developing computer science instructional mandates for elementary schools.

It is important that the teachers feel both confident and competent about that which they are teaching; teachers who are anxious about math teach math less (Geist, 2015). This study establishes initial evidence for a functional relationship between an analog approach and teacher confidence that is stronger than with a digital resource. More importantly, the group that interacted with the analog resource in this study was able to close the confidence gap between perceived and demonstrated competence. When developing state standards for computer science, it is recommended that policy makers adopt an analog-first instructional approach. This could include identifying play-based resources like Robot Turtles that present computational thinking concepts independent from electronic devices. While additional resource is necessary, the scaffolding aspect of a game such as Robot Turtles may be a critical element that supports the generation of greater teacher confidence as compared to less structured play-based resources. Finally, elementary level instruction in computer science should focus on the foundational
concepts of computational thinking as opposed to the direct application of programming in a specific language.

**Summary of recommendations.** The recommendations advanced here are primarily based around the transition from digital-first teaching of coding to an analog-based introduction of foundational concepts within computational thinking as a more effective method of implementing computer science in elementary classrooms. While digital instructional resources and practices can play a role in personalized learning for individual students, teacher-led instruction should be based on analog tools such as board games and the social discourse made possible by the group interactions around a tangible resource. This social discourse, along with the malleability of board games to meet intentional instructional objectives, are key components of success within the framework of cognitive acceleration. Professional development and classroom instruction need to change, but systemic change will only be possible if this approach is endorsed within standards, curriculum, and policies. Therefore, it is also essential that policy makers and companies providing resources for elementary schools be aware of this research and these recommendations.

**Limitations**

As with any study, this research was limited by certain factors. One limitation was the failure to retain the desired six subjects to allow for three cases in each group. After randomization was established but prior to consent, two of the subjects elected to withdraw from participation citing time constraints and, in one case, emerging health concerns. The potential impact of the two withdrawals was evaluated prior to continuance with the remaining four subjects. Recommended practices for single-case experimental
designs suggested the inclusion of at least three cases or tiers for evaluation between subjects and to demonstrate a functional or causal relationship (Shadish et al., 2014). While having only two cases per group was less than ideal, Vannest and Ninci (2015) noted that the judgement of the researcher in evaluating the data is also critical. Final determination of a functional relationship, Vannest and Ninci (2015) should be “based on a synthesis decision” looking at all aspects of the data set. Based on this assessment the decision was made to proceed with the study as planned with only four subjects. Statistical analysis was also limited by one subject not completing the necessary three baseline measurements prior to intervention.

**Conclusion**

Computer science instruction in elementary schools is an emerging topic. New state and national standards call for introduction of computer science in primary grades, yet there has been little research on best practice for teacher professional development at the elementary level. This study compared the impact of presenting the same professional development content to two groups of subjects who interacted with either an analog or digital resource during their learning. As a single-case experimental study, a multiple-baseline approach was used to establish control for each subject independently. Randomization techniques as recommended in the literature were also employed to increase the internal validity of the experimental design and support generalization of results and the identification of a functional relationship (Vannest & Ninci, 2015). Results were analyzed using both visual and statistical analysis following established procedures (Auerbach & Schudrich, 2013; Lane & Gast, 2014).
The results of this study indicated that the teachers who interacted with the analog resource, in this case the *Robot Turtles* board game, experienced statistically significant growth in their confidence as teachers of computational thinking. The growth in confidence for the analog group was greater than the confidence growth experienced by the digital group as indicated both visually and through comparison of effect sizes using Hedges’ *g*. These results corroborate prior research indicating a preference for analog interface modalities amongst adult females (Horn et al., 2009) as well as evidence for the efficacy of analog interfaces as compared to digital in terms of learning and application (Schneider et al., 2011). These findings add to the literature as some of the first empirical evidence of the efficacy of analog resources in computer science professional development. This was identified as a significant gap by Yaşar (2018). This study also provided recommendations for developing and implementing new practices implicated by these findings in terms of professional development, classroom instruction, and the development of policies and instructional standards. The crux of the recommendations is to adopt an analog-first approach to elementary computer science instruction based on computational thinking and implemented through the framework of cognitive acceleration.
References


Appendix A

Elementary Teacher Computer Programming Self-Efficacy Scale

This rating scale lists different instructional scenarios where you might be teaching about computer programming. You will be asked to rate how confident you are that you can complete these instructional tasks as of now. Rate your degree of confidence using the scale of 0 (Not at all Confident) to 10 (Highly Confident).

1. I can teach my students programming concepts using a tool like Scratch Jr. or Robot Turtles.
2. I can teach students how to identify a correct sequence of smaller steps to solve a larger programming task.
3. I can teach the use of If…Then…Else conditional programming statements.
4. I can teach the use of control flow in programming like loops or while statements.
5. I can teach about variables as a way to hold changing values of data.
6. I can teach students how to find repeating steps in a program that could become a function.
7. I can teach students how to package a section of code into a function to reuse it later.
8. I can teach about breaking down complex programming problems into smaller steps that can be more easily solved.
9. I can teach students to step through a program to identify and solve bugs.
10. I can teach about remixing existing code snippets into new programs to solve tasks.
11. I can teach students how to explain the reasoning behind their programming solution to a problem.
12. I can teach programming as an iterative process of incremental work to solve a problem.
13. I can teach my students how functions help make programming more efficient by addressing steps in a larger program.
14. I can teach students to identify computational problems in their daily lives.
15. I can teach students how to identify a computational problem and design a working solution to complete a task.

Appendix B

Adapted version of the Computational Thinking Test

Included here with permission from the original author, Román-González (2015).

The test was adapted for this study to use graphics from Robot Turtles and Scratch Jr.

Graphics from Robot Turtles including the turtle, gem, and cards used with permission.

Graphics from Scratch Jr. including the kitten, chicken, command blocks, and others used under the terms of The MIT License [https://opensource.org/licenses/MIT].

Copyright (c) 2016, Massachusetts Institute of Technology. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**EXAMPLE I**

Which instructions take the cat to the chicken by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
<th>![Option A Image]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option B</td>
<td>![Option B Image]</td>
</tr>
<tr>
<td>Option C</td>
<td>![Option C Image]</td>
</tr>
<tr>
<td>Option D</td>
<td>![Option D Image]</td>
</tr>
</tbody>
</table>
**EXAMPLE II**

*Which instructions take the turtle to the gem by the path marked out?*

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Option A" /></td>
<td><img src="image2" alt="Option B" /></td>
<td><img src="image3" alt="Option C" /></td>
<td><img src="image4" alt="Option D" /></td>
</tr>
</tbody>
</table>
Which instructions should the artist follow to draw the shape? The short side is one movement unit and the long side is two movement units.

<table>
<thead>
<tr>
<th>Option A</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Option A" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2" alt="Option B" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Option C" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4" alt="Option D" /></td>
</tr>
</tbody>
</table>
### QUESTION 1

*Which instructions take the cat to the chicken by the path marked out?*

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Option A" /></td>
<td><img src="image2" alt="Option B" /></td>
<td><img src="image3" alt="Option C" /></td>
<td><img src="image4" alt="Option D" /></td>
</tr>
</tbody>
</table>
### QUESTION 2

Which step is missing in the instructions below to take the cat to the chicken by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option A Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option B Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option C Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option D Image]</td>
</tr>
</tbody>
</table>

The instructions should take the turtle to the gem by the path marked out. In which step of the instructions is there a mistake?
QUESTION 4

Which instructions should the artist follow to draw the square? Each of the sides of the square measures one movement unit.

Option A

Option B

Option C

Option D
QUESTION 5

Which instructions take the cat to the chicken by the path marked out?

Option A

Option B

Option C

Option D
**QUESTION 6**

How many times must the sequence be repeated to take the cat to the chicken by the path marked out?

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Repeats</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>![Option A Image]</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>![Option B Image]</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>![Option C Image]</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>![Option D Image]</td>
</tr>
</tbody>
</table>
The instructions should make the artist draw the following rectangle **once** (one movement unit wide and two movement units high). In which step of the instructions is there a **mistake**?
**QUESTION 8**

Which instructions take the turtle to the gem by the path marked out?

- **Option A**
- **Option B**
- **Option C**
- **Option D**
QUESTION 9

Which instructions take the cat to the chicken by the path marked out?

Option A  
Option B

Option C  
Option D
QUESTION 10

Which step is missing in the instructions below to take the turtle to the gem by the path marked out?

Option A

Option B

Option C

Option D

Not missing any step
The instructions should take the cat to the chicken by the path marked out. In which step of the instructions is there a mistake?
### QUESTION 12

Which instructions should the artist follow to draw the ladder that reaches the flower?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Option A Image" /></td>
<td><img src="image2" alt="Option B Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Option C Image" /></td>
<td><img src="image4" alt="Option D Image" /></td>
</tr>
</tbody>
</table>
### QUESTION 13

Which instructions take the cat to the chicken by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Option A" /></td>
<td><img src="image2.png" alt="Option B" /></td>
</tr>
<tr>
<td>Option C</td>
<td>Option D</td>
</tr>
<tr>
<td><img src="image3.png" alt="Option C" /></td>
<td><img src="image4.png" alt="Option D" /></td>
</tr>
</tbody>
</table>
**QUESTION 14**

Which instructions take the turtle to the gem by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option A Image]</td>
<td>![Option B Image]</td>
<td>![Option C Image]</td>
<td>![Option D Image]</td>
</tr>
</tbody>
</table>

**Until**

**IF Path Right**

**THEN**

**Until**

**IF Path Right**

**THEN**

**Until**

**IF Path Right**

**THEN**

**Until**

**IF Path Left**

**THEN**
**QUESTION 15**

What is missing in the instructions below to take the cat to the chicken by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option A Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option B Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option C Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Option D Image]</td>
</tr>
</tbody>
</table>

Both Option A and Option C are Correct
The instructions should take the turtle to the gem by the path marked out. In which step of the instructions is there a mistake?
**QUESTION 17**

Which instructions take the turtle to the gem by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Option A Image" /></td>
<td><img src="image2.png" alt="Option B Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Option C Image" /></td>
<td><img src="image4.png" alt="Option D Image" /></td>
</tr>
</tbody>
</table>
QUESTION 18

Which instructions take the turtle to the gem by the path marked out?

Option A

Option B

Option C

Option D
The instructions should take the turtle to the gem by the path marked out. In which step of the instructions is there a mistake?
**QUESTION 20**

Which step is missing in the instructions below to take the turtle to the gem by the path marked out?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image A" /></td>
<td><img src="image2.png" alt="Image B" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Image C" /></td>
<td><img src="image4.png" alt="Image D" /></td>
</tr>
</tbody>
</table>

Not missing any step
**QUESTION 21**

Which instructions take the turtle to the ice wall by the path marked out and to tell the turtle to fire its laser to melt all the ice walls shown?

<table>
<thead>
<tr>
<th></th>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Option A" /></td>
<td><img src="image2" alt="Option B" /></td>
<td><img src="image3" alt="Option C" /></td>
<td><img src="image4" alt="Option D" /></td>
</tr>
</tbody>
</table>

- **Option A**: While Path Ahead x3
- **Option B**: While Path Ahead x4
- **Option C**: While Path Ahead x5
- **Option D**: While Path Ahead x3
### QUESTION 22

Which instructions take the turtle to the ice wall by the path marked out and to tell the turtle to fire its laser to melt all the ice walls shown?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Option A" /></td>
<td><img src="image2.png" alt="Option B" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Option C" /></td>
<td><img src="image4.png" alt="Option D" /></td>
</tr>
</tbody>
</table>
**QUESTION 23**

Which step is missing in the instructions below to take the cat to the fish by the path marked out and to tell the cat to eat all the fish shown?

<table>
<thead>
<tr>
<th>Option A</th>
<th>1 time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option B</td>
<td>2 times</td>
</tr>
<tr>
<td>Option C</td>
<td>3 times</td>
</tr>
<tr>
<td>Option D</td>
<td>5 times</td>
</tr>
</tbody>
</table>
**QUESTION 24**

Which step is missing in the instructions below to take the cat to the fish by the path marked out and to tell the cat to eat all the fish available?

<table>
<thead>
<tr>
<th>Option A</th>
<th>“While Path Ahead”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option B</td>
<td>“While No Path Ahead”</td>
</tr>
<tr>
<td>Option C</td>
<td>“While Any Fish”</td>
</tr>
<tr>
<td>Option D</td>
<td>“While No Fish”</td>
</tr>
</tbody>
</table>
**QUESTION 25**

The following function defined by the **blue message** draws a square, 10 units per side:

Which instructions should the artist follow to draw the following design?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Option A Diagram" /></td>
<td><img src="image2" alt="Option B Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Option C Diagram" /></td>
<td><img src="image4" alt="Option D Diagram" /></td>
</tr>
</tbody>
</table>
QUESTION 26

The following function defined by the blue message draws a square, 10 units per side:

The instructions below should make the artist draw the following design. What is missing in the instructions?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
The following set of instructions have been defined for the **function frog**:

Which instructions take the turtle to the ice walls by the path marked out and to tell the turtle to fire its laser to melt all the ice walls shown?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="x3" alt="Image" /></td>
<td><img src="x3" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="x5" alt="Image" /></td>
<td><img src="x5" alt="Image" /></td>
</tr>
</tbody>
</table>
**QUESTION 28**

The following set of instructions have been defined for the function frog:

What is missing in the instructions below to take the turtle to the ice walls by the path marked out and to tell the turtle to fire its laser to melt all the ice walls shown?

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C</th>
<th>Option D</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Appendix C

Permission to Use Computational Thinking Test

10/31/2017  sjf.edu Mail - Permission to use CTI

---

Harris, Christopher <cg@9070@sjf.edu>

---

Permission to use CTI
4 messages

Christopher Harris <cg@9070@sjf.edu> Fri, Aug 4, 2017 at 8:14 AM

To: Mrroman@edu.uned.es

Dr. Román-González,

I am a doctoral candidate at a small university in western New York State, USA. My dissertation topic involves preparing primary teachers to introduce computer science topics using a computational thinking approach. Specifically, I am looking at the potential impact using an analog or digital tool has on the confidence and competence of the teacher.

I saw your papers on the Computational Thinking Test and was very interested. I expect to use a multiple baseline experimental design so I have been looking for an instrument that would not be too difficult for subjects to complete multiple times.

Would it be possible to get permission to use an English language version of the CTI as an instrument to measure teacher competence for my research?

Thanks,
Christopher Harris

---

MARCOS ROMAN GONZALEZ <mrroman@edu.uned.es> Sat, Aug 5, 2017 at 7:12 AM

To: Christopher Harris <cg@9070@sjf.edu>
Cc: MARCOS ROMAN GONZALEZ <mrroman@edu.uned.es>

Hello Christopher,

Sure 😊... We're very pleased to share our CT-test for research purposes.

The easiest way to do it is through sharing a Google Form with the test in Google Drive...

...do you have a gmail account, please?

Best wishes,

-------------------

MARCOS ROMÁN GONZÁLEZ

Profesor e Investigador

Secretario Académico del Departamento de Métodos de Investigación y Diagnóstico en Educación I (MIDE I)

Facultad de Educación de la Universidad Nacional de Educación a Distancia (UNED)

C/ Juan del Rosal, nº14, Despacho 2.18. C.P. 28040. MADRID

Teléfono: 91 398 90 37

Web: http://goo.gl/oox5Qn

https://mail.google.com/mail/u/4/?ui=2&ik=d34924d4b&prev=/accounts/ARc1Ej54qwe7xyd8q1q8F988be56a07a&shld=132a92... 1/3
De: Christopher Harris <cgh09070@sjfc.edu>
Enviado: viernes, 4 de agosto de 2017 14:14
Para: MARCOS ROMAN GONZALEZ
Asunto: Permission to use CT1

[Quoted text hidden]

Christopher Harris <cgh09070@sjfc.edu>  Sat, Aug 5, 2017 at 8:38 AM
To: MARCOS ROMAN GONZALEZ <mroman@edu.uned.es>

Thank you for your prompt response. I am very excited to be using the CT-test as one of the very few validated measurements available. This is actually a mail account at my institution, so sharing here works.

Followup request: since I will be testing two types of interventions, one with Scratch (or Scratch JR) and the other with an analog board game called Robot Turtles, might it be possible for me to modify the graphics of the questions to depict the Robot Turtles tool that the teachers will use? I would take great care not to change the content or activity of the question, just the graphical overlay of the image. This would be done under the guidance of a dissertation committee member and would be submitted to you for review and approval prior to use. Our hope is that this would allow us to also assess whether there is any difference between the learning tool interface and the assessment interface.

Thank you,
Chris
[Quoted text hidden]

MARCOS ROMAN GONZALEZ <mroman@edu.uned.es>  Sun, Aug 6, 2017 at 6:49 PM
To: Christopher Harris <cgh09070@sjfc.edu>
Cc: MARCOS ROMAN GONZALEZ <mroman@edu.uned.es>

Hello Christopher,

About adapting the interface of the CT-test is not an easy decision...

...the CT-test reliability and validity have been already demonstrated; if you modify the test, even if the changes are limited to the graphics, then its psychometric properties might be violated...

...if you finally decide to make the adaptation, you must remember to:

- [ ]
- [ ]
- [ ]

...anyway, I guess this kind of decisions will be supervised by your tutors ;-) 

I've just shared the CT-test with you through Google Drive (You'll see it has an initial section named "Personal Information", that you can also edit to adapt it to your research needs). Besides, you can find attached:

- Our paper in CHB journal where the reliability and criterion validity of the test are reported.
- The CT-test in PDF and Word editable formats, including the correct answers.
- An Excel file with the items specifications.

Keep in touch about your progress please 😊

Best wishes
MARCOS ROMÁN GONZÁLEZ

Profesor e Investigador

Secretario Académico del Departamento de Métodos de Investigación y Diagnóstico en Educación I (MIDE I)
Facultad de Educación de la Universidad Nacional de Educación a Distancia (UNED)
C/ Juan del Rosal, nº14. Despacho 2.18. C.P. 28040. MADRID
Teléfono: 91 398 90 37
Web: http://goc.gl/oox5Qn

De: Christopher Harris <cgh09070@sjfc.edu>
Enviado: sábado, 5 de agosto de 2017 14:38
Para: MARCOS ROMÁN GONZÁLEZ
Asunto: Re: Permission to use CTt

[Quoted text hidden]

4 attachments

TPC (english_version) - final.docx
358.8K

TPC (english_version) - final.pdf
276.1K

1-s2.0-S0747563216306185-main.pdf
17.91K

Especificacion_Items_TPC (v 2.0) - ENGLISH.xlsx
14K
Hi Christopher,

Thank you very much for the feedback.

Your results and implications are really interesting for us 😊

Yes, you have my permission to include the whole CTest in your dissertation. Good luck with your thesis defense!!!

Best wishes from Spain,

-------------------------------
MARCOS ROMÁN GONZÁLEZ

Profesor e Investigador

Secretario Académico del Departamento de Métodos de Investigación y Diagnóstico en Educación I (MIDE I)

Facultad de Educación de la Universidad Nacional de Educación a Distancia (UNED)

C/ Juan del Rosal, nº14, Despacho 2.18. C. P. 28040. MADRID

Teléfono: 91 398 90 37

Web: http://go.gl/loxx5Qn
Appendix D

First Professional Development Session
Computational Thinking

Computational thinking refers to a set of skills and concepts that are foundational to computer science, but transferable to other disciplines. Instead of focusing on a particular programming language, computational thinking introduces basic concepts found in almost every computer language such as if-then statements. International studies have shown that mastery of computational thinking is correlated with higher levels of success in core academic subject tests. This isn’t surprising. Computational thinking is about learning how to examine tasks, ask questions, and solve problems.

In these professional development sessions, we will be looking at computational thinking through the lens of Piaget and Vygotsky. The purpose is to create a developmentally appropriate approach to computer science instruction in primary grades. From Piaget, we draw the need for a concrete approach to a new subject. You will see this in the use of kinesthetic movement for programming commands. Vygotsky’s work provides the concept of accelerated development through social interactions and teacher support. The goal of this program is to help you, the teacher, become more confident and competent with computational thinking so that you can best support your students as learners.

Lesson One: Robotic Programming

A great way to get started with computational thinking is through robotics. Children are accustomed to directing the movement of toys in their play, and so the idea of giving movement commands to a robot or character is familiar. But don’t think that you need an expensive robot to get started.

Vocabulary

Command - telling the robot what you want it to do.
Relative Directions - movement commands are given to the robot that will be executed relative to the robot’s current position and the direction it is facing.
Absolute Directions - movement commands are given in absolute terms that include the directionality of the movement such as go North.

Giving Commands

Robot Turtles is a board game that introduces students to the concept of giving directional commands to a robot. The robots in this game use relative directions meaning that their movement is relative to their position on the board. Instead of a command to go “North” Robot Turtle has a card that just tells the robot to go “Forward” in the direction it is already facing. If you want the robot to move in a different direction, you first have to turn it using the arrow cards, and then direct it to move forward in the direction it is facing. Other robots use absolute directions.

Longer chains of commands can be created to send the robot a series of directions all at once. But be careful! A wrong turn at the start of a direction chain can send the robot in an unexpected direction! Any mistakes will need to be debugged.

Getting Started

To introduce students to the idea of giving and receiving movement commands start with a kinesthetic movement activity in which the students are the robots being programmed. You can hold up movement command cards similar to those found in Robot Turtles to show students how to move. This works especially well in a classroom or hallway with square floor tiles. Make sure that students understand the concept of relative directions. In Robot Turtles, the turn commands are just turning; there is no movement.

Lesson 2 - If/Then and Control Flow Statements

Once students have mastered the basics of movement, you can introduce the first bit of computer programming. Control flow statements are a collection of computer programming commands that direct actions within a program. These types of commands are found in every language even if the actual name of the command sometimes changes.

Vocabulary

Control Flow - a programming concept that helps direct the flow of a computer through a program by describing decision points.

The most basic, and one of the most powerful, control flow statements is If/Then. We actually use If/Then statements every day as we think about decisions and tell people what to do. “It’s raining; take an umbrella.” We already teach If/Then as part of ELA instruction when we talk about cause and effect. Developmental psychologists also refer to this as stimulus/response. This is nothing new, just a new way of thinking about giving a command.

In Robot Turtles we can introduce If/Then statements using the Ice Walls. If there is an ice wall in front of our turtle, then we need to fire the laser to melt the wall.

Classroom Connection

You can practice If/Then statements in the classroom by being highlighting the usage in your directions. Or, combine If/Then statements into the kinesthetic movement activity described above - if you are wearing blue, then move forward.
Computational Thinking

Computational thinking refers to a set of skills and concepts that are foundational to computer science, but transferable to other disciplines. Instead of focusing on a particular programming language, computational thinking introduces basic concepts found in almost every computer language such as if-then statements. International studies have shown that mastery of computational thinking is correlated with higher levels of success in core academic subject tests. This isn’t surprising. Computational thinking is about learning how to examine tasks, ask questions, and solve problems.

In these professional development sessions, we will be looking at computational thinking through the lens of Piaget and Vygotsky. The purpose is to create a developmentally appropriate approach to computer science instruction in primary grades. From Piaget, we draw the need for a concrete approach to a new subject. You will see this in the use of kinesthetic movement for programming commands. Vygotsky’s work provides the concept of accelerated development through social interactions and teacher support. The goal of this program is to help you, the teacher, become more confident and competent with computational thinking so that you can best support your students as learners.

Lesson One: Robotic Programming

A great way to get started with computational thinking is through robotics. Children are accustomed to directing the movement of toys in their play, so the idea of giving movement commands to a robot or character is familiar. But don’t think that you need an expensive robot to get started.

Vocabulary

Command - telling the robot what you want it to do.
Relative Directions - movement commands are given to the robot that will be executed relative to the robot’s current position and the direction it is facing.
Absolute Directions - movement commands are given in absolute terms that include the directionality of the movement such as go North.

Giving Commands

Scratch Jr. is a programming app that introduces students to the concept of giving directional commands to a robot. In this case, our kitten sprite, uses absolute directions. This means that the movement is always the same for the command given. If you give the kitten a move up command, it will appear to jump straight up on the screen. Move backwards will cause it to slide backards. It doesn’t matter what direction the kitten is facing, the response to the movement command is the same. Other robots use relative directions where the position of the robot matters for movement.

Longer chains of commands can be created to send the robot a series of directions all at once. But be careful! A wrong turn at the start of a direction chain can send the robot in an unexpected direction! Any mistakes will need to be debugged.

Getting Started

To introduce students to the idea of giving and receiving movement commands start with a kinesthetic movement activity in which the students are the robots being programmed. You can hold up movement command cards similar to those found in Scratch Jr. to show students how to move. This works especially well in a classroom or hallway with square floor tiles. Make sure that students understand the concept of absolute directions. In Scratch Jr, the commands are executed as given without any regard for the direction the student is facing at the time of the command.

Lesson 2 - If Then and Control Flow Statements

Once students have mastered the basics of movement, you can introduce the first bit of computer programming. Control flow statements are a collection of computer programming commands that direct actions within a program. These types of commands are found in every language even if the actual name of the command sometimes changes.

Vocabulary

Control Flow - a programming concept that helps direct the flow of a computer through a program by describing decision points.

The most basic, and one of the most powerful, control flow statements is If Then. We actually use If Then statements every day as we think about decisions and tell people what to do. "If it is raining, then take an umbrella." We already teach If Then as part of ELA instruction when we talk about cause and effect. Developmental psychologists also refer to this as stimuli’s response. This is nothing new, just a new way of thinking about giving a command.

In Scratch Jr, we can introduce If Then statements using the Start on Bump block. If the kitten touches another character, then do something.

Classroom Connection

You can practice If Then statements in the classroom by highlighting the usage in your directions. Or, combine If Then statements into the kinesthetic movement activity described above – if you are wearing blue, then move forward.
Appendix E

Second Professional Development Session
Computational Thinking

Lesson Three: Putting Things Together
In the first lesson, we looked at giving basic commands to a robot as a way of practicing the skills and concepts of computational thinking. Some systems use relative direction commands while others use absolute directions. Either way, the commands given need to be precise and specific for the computer to follow them.

We also looked at a basic building block that is found in almost all computer programming languages, the If/Then control flow statement. If/Then is a decision point that directs the computer towards a specific action (then) when a condition is met (if). In this second lesson, we are going to look at putting commands and control flow statements together into packages.

Vocabulary
Control Flow - a type of command that tells the computer what to do given a certain situation.
Algorithm - like a recipe for baking, an algorithm is a precise and specific set of directions for solving a problem or completing a task.
Function - also called routines in some languages, this is a way of packaging up a set of commonly used commands to be repeatedly called with a single command.

Control Flow
If/Then is incredibly powerful. It lets you, the computer programmer, pre-define conditions and responses that will then allow the computer to proceed through a process without needing additional input. If the space the robot is on is red Then turn right. The robot can check for the condition and respond. Once.

The limitation of If/Then is that it only checks a single point in time. To overcome this limitation, computer programs often use a more complicated form of control flow called looping. Loops run repeated checks for conditions, like repeated If/Then statements, so that decision points can happen more than once. Different languages use different terms for loops, but there are two main styles of loops depending on if you need a set number of loops or an unknown number of loops.

Functions
These packaged sets of commands are called functions (or sometimes routines). A function creates a tidy package that gives a series of commands a single name. This lets a savvy programmer call, or execute, that series of commands as needed throughout a longer program. In Robot Turtles, the function ‘get ready for bed’ represents functions by carrying the turtle robot through a series of commands.

Functions are defined by creating a name for the package and then describing the commands that will be included in the package. We use functions every day without even thinking about them. For example, if you tell a child to “get ready for bed” both of you understand that there are multiple steps that must be taken. “Get ready for bed” is a function that includes a series of commands like: 1) take a shower 2) put on pajamas 3) brush your teeth and 4) get tucked in bed. But instead of having to give all four commands every night, you can simply call the function that includes all the steps.

Algorithms
Functions are often included within algorithms, precise and specific directions for solving a problem or completing a task. Algorithms really should be almost pain fully precise and specific. Computers cannot infer, so programmers have to give directions that cannot be misinterpreted or misunderstood. A classic exercise to demonstrate an algorithm is to have children give you directions for making a PB&J. If the first command is to spread the peanut butter on the bread, then you pick up the jar of peanut butter and rub it on the loaf of bread in the bag.

Functions can help make these precise and specific algorithms easier to describe by letting the programmer re-use chunks of code. In the PB&J example, you could define a function “spread” that includes commands for dipping the knife into a substance to coat and then spreading the substance on the bread using a sweeping gesture. The “spread” function could then be re-used for both the peanut butter and the jelly. This is a great activity for even primary students, as it makes them think.

Pseudocode
When you are describing actions as algorithms, using control flow and functions, what you are really doing is writing a simple form of computer programming called pseudocode. Pseudocode uses plain English, but with a heavy dose of structure and precision, to convey the underlying idea of a computer program without using code. Pseudocode is a great way to describe what you want an app or program to do so you can explain it to someone who can help you write the code for it. Try this out!
Computational Thinking

Lesson Three: Putting Things Together

In the first lesson, we looked at giving basic commands to a robot as a way of practicing the skills and concepts of computational thinking. Some systems use relative direction commands while others use absolute directions. Either way, the commands given need to be precise and specific for the computer to follow them.

We also looked at a basic building block that is found in almost all computer programming languages, the If/Then control flow statement. If/Then is a decision point that directs the computer towards a specific action (then) when a condition is met (if). In this second lesson, we are going to look at putting commands and control flow statements together into packages.

Vocabulary

Control Flow - a type of command that tells the computer what to do given a certain situation.
Algorithm - like a recipe for baking, an algorithm is precise and specific set of directions for solving a problem or completing a task.
Function - also called routines in some languages, this is a way of packaging up a set of commonly used commands to be repeatedly called with a single command.

Advanced Control Flow

If/Then is incredibly powerful. It lets you, the computer programmer, pre-define conditions and responses that will then allow the computer to proceed through a process without needing additional input. If the space the robot is on is red Then it's right. The robot can check for the condition and respond. Once.

The limitation of If/Then is that it only checks a single point in time. To overcome this limitation, computer programs often use a more complicated form of control flow called looping. Loops run repeated checks for conditions, like repeated If/Then statements, so that decision points can happen more than once. Different languages use different terms for loops, but there are two main styles of loops depending on you need a set number of loops or an unknown number of loops.

The Repeat or For loop is used when you know how many times you want to loop your check. For 5 times do this. If the number of loops is unknown, but an ending condition can be described, you can use a While or Until loop. If it is raining then open your umbrella describes an action for a single point in time with no rule for closing the umbrella. While it is raining, open your umbrella tells the robot to open the umbrella when it starts raining and keep it open for the duration of the rain.

Functions

These packaged sets of commands are called functions (or sometimes routines). A function creates a tidy package that gives a series of commands a single name. This lets a savvy programmer call or execute, that series of commands as needed throughout a longer program. In Scratch Jr, envelopes represent the packaging and sending of commands by a function.

Functions are defined by creating a name for the package and then describing the commands that will be included in the package. We use functions every day without even thinking about them. For example, if you tell a child to “get ready for bed” both of you understand that there are multiple steps that must be taken. "Get ready for bed" is a function that includes a series of commands like 1) take a shower 2) put on pajamas 3) brush your teeth and 4) get tucked in to bed. But instead of having to give all four commands every night, you can simply call the function that includes all the steps. Or, in the case of Scratch Jr, send that colored envelope message.

Algorithms

Functions are often included within algorithms, precise and specific directions for solving a problem or completing a task. Algorithms really should be almost pain full and precise. Computers can not infer, and so programmers have to give directions that cannot be misinterpreted or misunderstood. A classic exercise to demonstrate an algorithm is to have children give you directions for making a PB&J. If the first command is to spread the peanut butter on the bread, then you pick up the jar of peanut butter and rub it on the loaf of bread in the bag.

Functions can help make these precise and specific algorithms easier to describe by letting the programmer re-use chunks of code. In the PB&J example, you could define a function “spread” that includes commands for dipping the knife into a substance to coat and then spreading the substance on the bread using a sweeping gesture. The “spread” function could then be re-used for both the peanut butter and the jelly. This is a great activity for even primary students, as it makes them think.

Pseudocode

When you are describing actions as algorithms, using control flow and functions, what you are really doing is writing a simple form of computer programming called pseudocode. Pseudocode uses plain English, but with a heavy dose of structure and precision, to convey the underlying idea of a computer program without using code. Pseudocode is a great way to describe what you want an app or program to do so you can explain it to someone who can help you write the code for it. Try this out!
Appendix F

Competence Results by Computational Thinking Concept for Baseline (B) and Intervention (I) Phases

**Competence Results by Concept Addressed**

<table>
<thead>
<tr>
<th>Case</th>
<th>Overall M</th>
<th>Directions M</th>
<th>Repeat For M</th>
<th>Until M</th>
<th>If-Then M</th>
<th>If-Else M</th>
<th>While M</th>
<th>Function M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>B</td>
<td>I</td>
</tr>
<tr>
<td>Jumper</td>
<td>67.14</td>
<td>70.24</td>
<td>100</td>
<td>100</td>
<td>70</td>
<td>66.67</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>jlo</td>
<td>58.57</td>
<td>70.24</td>
<td>70</td>
<td>95.83</td>
<td>60</td>
<td>79.17</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>Taylor</td>
<td>70.54</td>
<td>68.75</td>
<td>100</td>
<td>100</td>
<td>68.75</td>
<td>75</td>
<td>56.25</td>
<td>4.03</td>
</tr>
<tr>
<td>Charlotte</td>
<td>58.93</td>
<td>65.18</td>
<td>93.75</td>
<td>93.75</td>
<td>68.75</td>
<td>68.75</td>
<td>62.50</td>
<td>3.76</td>
</tr>
</tbody>
</table>

Note: B = baseline phase, I = intervention phase
Appendix G

Permission letter for use of Robot Turtles graphics.
July 12, 2018

Christopher Harris
St. John Fisher College
3690 East Ave
Rochester, NY 14618

RE: Permission of use Robot Turtles™

Dear Mr. Harris,

We are responding to your request to obtain permission to include images from the game Robot Turtles™ in your dissertation work for St. John Fisher College. Your request, as stated in the June 27, 2018 email, is to "use graphics including some of the cards, turtles, gems, and a page from the rule book within (the) dissertation and in the assessment instrument being created to measure teacher growth. The dissertation would be electronically published as a PDF document on the St. John Fisher website and made available for free access by other researchers. The assessment would be included as an appendix for informational purposes only and not intended for reuse (it is adapted from another researcher's work)". We are pleased to inform you that we are granting permission for you to use the images for the purpose stated in this letter.

All material provided in the game Robot Turtles™ is the intellectual property of ThinkFun, and we are sure that you will understand our need to protect the use of it. As such, we would like to receive acceptance to the following requirements:

1. The permission is granted only to the request of using the images in your dissertation, to be published as a PDF document on the St. John Fisher website.
2. The permission is granted to use the images of the games in your dissertation, however, it is understood that there will be no charge for distribution of the images.
3. Please include the TM & © symbols and company legal lines when referencing the artwork in your publication as follows:

   Robot Turtles™
   Copyright 2014 Robot Turtles, LLC.
   © 2014 ThinkFun, Inc.

At ThinkFun, one of our core beliefs is that learning and stretching your brain can be fun! Over 30 years ago our company was founded with the dreams of changing the world through play, and whether you're 5 years old or 90 years old, we are firm believers that games can sharpen your mind and build life-long skills. We are pleased that you have chosen our games for your doctoral research.

Please sign the letter below and either email (kimberly@thinkfun.com) or fax (703-549-6210) it back to us. Let me know if you have any questions. I look forward to hearing back from you.

Best regards,

Kimberly McKenna
Jr. Product Development Manager
ThinkFun Inc.

[Signature]

ThinkFun Signature, Date

Christopher Harris, Date