AKSIT, OSMAN. Enhancing Science Learning through Computational Thinking and Modeling in Middle School Classrooms: A Mixed Methods Study. (Under the direction of Dr. Eric N. Wiebe).

Computational thinking and modeling are authentic practices that scientists and engineers use frequently in their daily work. Advances in modern computing technologies over the past couple of decades have further emphasized the centrality of modeling in science by making computationally-enabled model use and construction more accessible to scientists. For this reason, it is important for students to get exposed to these practices in K-12 science classrooms. This mixed-methods study investigated how a one-week intervention in regular middle school science classrooms that introduced computational thinking practices and simulation-based model building through a visual block-based programming environment influenced students’ understanding of computational thinking concepts and practices as well as their conceptual understanding of a physical science topic, force and motion. The study also examined whether and how seventh-grade students’ attitudes towards programming changed as a result of participating in the intervention course. Eighty-two seventh-grade students from a public middle school in the southeastern United States participated in the study. Quantitative data sources included pre- and post-tests of students’ knowledge of computational thinking concepts and practices, force and motion concepts, and their attitudes towards programming. Qualitative data sources included classroom observation notes, interviews with students, and reflection statement responses. During the classroom intervention, students were introduced to computational thinking concepts and practices through a block-based programming environment called Scratch and constructed simulation-based computational models of physical phenomena. The findings of the study indicated that...
engaging in building computational models through Scratch programming environment in regular science classrooms resulted in significant conceptual learning gains for the sample of this study. The affordances of the dynamic nature of computational models let students both “observe” and “interact” with the target phenomenon in real time while the generative dimension of model construction promoted a rich classroom discourse facilitating conceptual learning. The results also indicated that students had more favorable attitudes towards programming on the post-test compared to the pre-test. This study contributes to the nascent literature on integrating computational modeling into K-12 science curricula by emphasizing the affordances and generative dimension of model construction through block-based programming.
Enhancing Science Learning through Computational Thinking and Modeling in Middle School Classrooms: A Mixed Methods Study

by
Osman Aksit

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Science Education

Raleigh, North Carolina

2018

APPROVED BY:

__________________________________________________________________________
Dr. Eric N. Wiebe
Committee Chair

__________________________________________________________________________
Dr. M. Gail Jones

__________________________________________________________________________
Dr. Cesar Delgado

__________________________________________________________________________
Dr. Karen S. McNeal
DEDICATION

I dedicate this dissertation to my wife, Dilek Aksit, for her unconditional support at every stage of my doctoral studies.
BIOGRAPHY

Osman Aksit was born and raised in Izmir, Turkey. After finishing high school, he moved to Istanbul to attend Bogazici University where he earned his bachelor’s degree in Physics Education. After graduation, he started working as a physics teacher at Bilkent Laboratory School in Erzurum. While teaching at Bilkent, he continued his professional development and earned his master’s degree from Bogazici University and became interested in doing educational research on science learning with K-12 students. After working as a physics teacher for three years, he decided to further continue his graduate studies and started his Ph.D. in Science Education at North Carolina State University in 2014.
ACKNOWLEDGMENTS

First and foremost, I am very grateful to my committee chair and advisor, Dr. Eric N. Wiebe, for making this journey possible and enjoyable. His valuable guidance as my advisor and mentor during my doctoral studies has been essential for my scholarly growth. I would also like to acknowledge my committee members, Dr. M. Gail Jones, Dr. Cesar Delgado, and Dr. Karen S. McNeal, for their feedback and support throughout my dissertation research. Last but not least, I would like to express my appreciation to my fellow graduate students who shared this journey with me: Alonzo Alexander, Kylie Swanson, Kemal Akoglu, Megan Ennes, Fulya Eyupoglu, Katherine Chesnutt, Emily Cayton, Katie Green, Kristie Gutierrez, Rebecca Hite, Whitney Richardson, James Creager, Bita Akram, and Danielle Boulden. A special thanks goes to Cody Smith who helped me tremendously by serving as a second coder in this study.
TABLE OF CONTENTS

LIST OF TABLES ....................................................................................................................... ix

LIST OF FIGURES ..................................................................................................................... xi

CHAPTER ONE: INTRODUCTION ................................................................................................. 1

Theoretical Frameworks ........................................................................................................... 9

Constructionism ......................................................................................................................... 9

Learning through Constructing Representations ................................................................. 11

Purpose of the Study ............................................................................................................... 13

Summary ................................................................................................................................. 14

CHAPTER TWO: REVIEW OF THE LITERATURE ......................................................................... 16

Historical Background of Computational Thinking ............................................................ 16

Defining Computational Thinking ......................................................................................... 18

Computational Thinking Practices ......................................................................................... 21

Computational Thinking in K-12 STEM Education ............................................................ 25

Assessment of Computational Thinking ............................................................................... 27

Models and Modeling in Science ......................................................................................... 29

Models and Modeling in Science Education ....................................................................... 30

Computational Thinking and Modeling ............................................................................... 32

Learning and Reasoning with Representations ................................................................. 33
Pre-test/Post-test. ........................................................................................................... 64
Student Interviews. ......................................................................................................... 69
Reflective Statement Responses. ..................................................................................... 77
Research Question 2........................................................................................................ 79
Pre-test/Post-test. ........................................................................................................... 79
Student Interviews. ......................................................................................................... 83
Reflective Statement Responses. ..................................................................................... 85
Research Question 3........................................................................................................ 88
Pre-test/Post-test. ........................................................................................................... 88
Student Interviews. ......................................................................................................... 93
Reflective Statement Responses. ..................................................................................... 94
Summary .......................................................................................................................... 96

CHAPTER FIVE: DISCUSSION AND IMPLICATIONS ......................................................... 97
Research Question 1....................................................................................................... 97
Research Question 2....................................................................................................... 111
Research Question 3....................................................................................................... 120
Limitations of the Study.................................................................................................. 127
Conclusion and Future Work ......................................................................................... 129
REFERENCES ............................................................................................................... 132
APPENDICES ............................................................................................................. 153

Appendix A ............................................................................................................. 154

Appendix B ............................................................................................................. 161

Appendix C ............................................................................................................. 160

Appendix D ............................................................................................................. 154

Appendix E ............................................................................................................. 157
LIST OF TABLES

Table 3.1  Computational thinking practices and modeling activities covered during the intervention .................................................................48
Table 3.2  State Science Standards addressed during the modeling activities ..............48
Table 3.3  A sample question from the Force and Motion Assessment ..............................55
Table 3.4  The list and descriptions of the data sources ..............................................59
Table 4.1  Descriptive statistics for students’ scores on the Force and Motion Assessment .................................................................................................65
Table 4.2  Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the Force and Motion Assessment ..............................................67
Table 4.3  Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the Force and Motion Assessment by gender ........................................68
Table 4.4  Results of the independent-samples t-test analysis comparing the change in pre- and post-test scores for the Force and Motion Assessment between male and female students .................................................................................................69
Table 4.5  Themes from student interview responses regarding their conceptual understanding of force and motion concepts ............................70
Table 4.6  Themes from student interview responses regarding their understanding and use of models .................................................................................................72
Table 4.7  Themes from student interview responses regarding their conceptual understanding of the nature of models in science ............................74
Table 4.8  Themes from student interview responses regarding their views on how models that they built helped them better understand force and motion concepts .................................................................................................76
Table 4.9  Themes from student reflective statement responses regarding their conceptual understanding of force and motion concepts ..............78
Table 4.10 Descriptive statistics for students’ scores on the CTt ........................................80
Table 4.11 Results of the paired-samples t-test comparing students’ pre- and post-test scores on the CTt .................................................................82

Table 4.12 Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the CTt by gender .................................................................82

Table 4.13 Results of the independent-samples t-test analysis comparing the change in pre- and post-test scores for the CTt between male and female students ........83

Table 4.14 Themes from student reflective statement responses regarding their understanding of computational thinking concepts and practices ..................86

Table 4.15 Descriptive statistics for students’ scores on the attitudinal survey ........89

Table 4.16 Descriptive statistics for students’ average scores on each survey item ..........91

Table 4.17 Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the attitudinal survey .................................................................92

Table 4.18 Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the attitudinal survey by gender .................................................................92

Table 4.19 Results of the independent-samples t-test analysis comparing the change in pre- and post-test scores for the attitudinal survey between male and female students .................................................................93

Table 4.20 Quotes from student interview responses regarding their positive attitudes towards programming .................................................................94

Table 4.21 Themes from student reflective statement responses regarding their attitudes towards programming .................................................................96
LIST OF FIGURES

Figure 1.1. Block-based versus traditional text-based programming ........................................7

Figure 3.1. Convergent mixed methods design of the study .......................................................43

Figure 3.2. User interface of the Scratch programming environment ....................................46

Figure 3.3. Screenshot of the first modeling activity in Scratch .................................................50

Figure 3.4. Screenshot of the second modeling activity in Scratch ...........................................51

Figure 3.5. A sample question from the CTt ..............................................................................54

Figure 4.1. Histogram of the pre-test scores of the Force and Motion Assessment .................66

Figure 4.2. Histogram of the post-test scores of the Force and Motion Assessment ..............66

Figure 4.3. Histogram of the pre-test scores of the CTt ............................................................80

Figure 4.4. Histogram of the post-test scores of the CTt ...........................................................81

Figure 4.5. Histogram of the pre-test scores of the Attitudinal Survey ................................89

Figure 4.6. Histogram of the post-test scores of the Attitudinal Survey ................................90

Figure 5.1. Mathematical formula versus block-based code on Scratch .............................108

Figure 5.2. Calculating velocity using block-based code .........................................................109

Figure 5.3. A sample pseudocode for a game algorithm (adapted from Davies, 2008) ......118
CHAPTER ONE: INTRODUCTION

The advancement of computing and computer science has greatly influenced how science is practiced by scientists today. This has resulted in computational thinking becoming an increasingly important skill and core competency in all science, technology, engineering, and mathematics (STEM) fields (National Research Council [NRC], 2010). Scientists today use computer technologies to conduct experiments, create complex models of physical phenomena, collect and analyze very large data sets, and identify patterns and relationships in a rapid and efficient way that was not possible in the past without computers. The Next Generation Science Standards (NGSS) identifies computational thinking as one of the eight science and engineering key practices that are essential for all students to master in K-12 education (NGSS Lead States, 2013). A growing body of studies has shown that the effective use of computational thinking concepts and practices in science classrooms can support and promote conceptual science learning (Basu et al., 2016; diSessa, 2000; Sengupta et al., 2013; Weintrop et al., 2016; Wilensky, Brady, & Horn, 2014).

Computational thinking has received extensive attention from a broad range of communities including computer scientists, educational researchers, disciplinary experts, policymakers, and K-12 educators over the last decade. The term “computational thinking” was first introduced by Wing (2006) to refer to “a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use” (p. 33). Although there is a debate in the scholarly community on the definition of computational thinking, and what specific concepts, skills and practices computational thinking involves (Grover & Pea, 2013; Weintrop et al., 2016), researchers and policymakers agree on the idea that...
computational thinking is a necessary skill to acquire for all students (Barr & Stephenson, 2011; K-12 Computer Science [CS] Framework, 2016; NRC, 2012; Wing, 2014). As computational tools and technologies have become an important part of everyday life in the 21st century, computational thinking can help students become not just consumers but also producers of technology by fostering creativity (Mishra & Yadav, 2013).

Computational thinking can be simply defined as the “thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms” (Aho, 2012, p. 832). This definition draws on the idea that computational solutions are produced in forms that can be executed by an “information-processing agent” (Cuny, Snyder, & Wing, 2010). Computational thinking requires an understanding of the potential and limitations of computers, approaching and formulating problems that can be addressed by computing, and developing abstractions and algorithms that can be carried out by computers (K-12 CS Framework, 2016). Barr and Stephenson (2011) identified nine core concepts and practices pertaining to computational thinking which include data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation.

Despite the apparent consensus on computational thinking as a key set of concepts and practices central to STEM disciplines, research shows that computational thinking has not been well integrated into K-12 science curricula nor addressed properly in science classrooms (Sengupta et al., 2013; Voogt, Fisser, Good, Mishra, & Yadav, 2015; Wilensky et al., 2014). Across the STEM secondary school course offerings, students lack access to
formal instruction in computational thinking (Wilson, 2013). For example, only about 12% of high schools nationwide offer AP Computer Science Principles course (College Board, 2016). Likewise, middle schools in the United States face a significant shortage of teachers to teach introductory computer science which can help foster learning about computational thinking across subject areas (Grover, Pea, & Cooper, 2015). Existing R&D efforts have mostly focused on introducing computational thinking in open-ended contexts including game design and storytelling with programming, or in informal settings through extracurricular activities (e.g., robotics; Bers, 2010; Bers, Flannery, Kazakoff, & Sullivan, 2014; Denner, Werner, & Ortiz, 2012).

Embedding computational thinking concepts and practices in science classrooms can serve to both expose all students in a school to computational thinking and enrich authentic science learning (Wilensky et al., 2014; Weintrop et al., 2016). However, the idea of using computational thinking as a vehicle to directly facilitate and foster science learning in formal classroom settings is under-investigated in the literature (Grover & Pea, 2013; Basu et al., 2016). Similarly, little is known about how to best integrate computational thinking practices into K-12 science curricula and instruction in ways that can support both computational thinking and science learning in the classroom (Basu et al., 2016; Sengupta et al., 2013). A number of researchers have suggested that modeling—which is also a key science practice—can be used as an effective medium to introduce and promote computational thinking in addition to furthering science learning in formal classroom settings (Basu et al., 2016; Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009; Sengupta et al., 2013; Weintrop et al., 2016; Wilensky et al., 2014).
Science education researchers have long argued that science learning should not only focus on learning well-established scientific theories, discoveries, and facts but also engage students in authentic practices that scientists use in their professions (Marx et al., 2004; NRC, 1996). Specifically, the practice of modeling has not been incorporated extensively into K-12 science curricula although it is an essential part of the scientific inquiry today (Lehrer & Schauble, 2006; Schwarz et al., 2009). Similar to computational thinking, modeling is one of the eight science and engineering key practices identified by the NGSS (Lead States, 2013). Modeling can be defined as the practice of creating a simplified, external representation of a system to explain a scientific concept such as the particle model of matter (Schwarz et al., 2009), and can help students link observed scientific phenomena and underlying abstract conceptual underpinnings (Harrison & Treagust, 1998; Leenaars, van Joolingen, & Bollen, 2013). Previous research suggests that students who engage in developing and using models end up having a more robust understanding of natural phenomena (Clement, 2000; Oh & Oh, 2011).

A scientific model can be a conceptual, mathematical or physical representation of natural phenomena. Models are built by using a variety of tools such as plots, diagrams, physical structures, computer programs, and simulations (Bailer-Jones, 2013). Different representational tools have different affordances. For instance, models built through diagrams, pictures, and drawings—although useful in describing the spatial relationships between the different components of a system—are inherently static representations that do not illustrate how the system would behave or change over time. However, simulation-based computer models are dynamic representations that can show how the target system or
physical phenomenon would behave or change under different conditions in real time (Boulter & Buckley, 2000; Delgado & Krajcik, 2010). Constructing simulation-based models requires students to articulate on their understanding of different variables included in defining a system, and explicitly identifying the relationships (i.e., the causal mechanisms) among these variables (de Jong & van Joolingen, 1998; diSessa, 2000; Papert, 1980). Previous research has shown that engaging students in computational thinking through computer programming to build simulation-based models that explain and predict the behavior of physical phenomena can be an effective modeling strategy which can foster both computational thinking and science learning in K-12 science classrooms (Klopfer et al., 2009; Sengupta et al., 2013; Sherin, diSessa, & Hammer, 1993; Wilensky & Resnick, 1999; Wilensky et al., 2014).

Although computers and computational tools are becoming increasingly ubiquitous in today’s classrooms, the use of computers as a medium of inquiry in science classrooms has been very limited. Since students do not have much exposure to computational thinking and computer programming in K-12, engaging in developing models through programming can be particularly challenging for students (Basu et al., 2016; Foley, 2012). First of all, in order to build computer models of scientific phenomena, students need to have both conceptual understanding (e.g., conditionals, loops, algorithms) and operational knowledge (e.g., user interface, syntax, semantics) of a particular programming language, in addition to the domain knowledge of the science topic being studied. Previous research suggests that learning programming is difficult for students of all ages (Kelleher & Pausch, 2005; Mannila, Peltomäki, & Salakoski, 2006; McCracken et al., 2001). Learning about syntax and
semantics of a programming language is one of the common and most frustrating challenges for new learners (Guzdial, 2004; Kelleher & Pausch, 2005; Mannila et al., 2006). Specifically, learners usually struggle with "remembering the names of commands, the order of parameters, whether or not they are supposed to use parentheses or braces, and so on" (Kelleher & Pausch, 2005, p. 90). In order to address the challenges related to learning syntax and semantics, and also make it easier for novice programmers to focus on the essential programming concepts without getting distracted by the mechanics of programming, a number of visual block-based programming environments have been developed to serve as introductory tools for programming (Shapiro & Ahrens, 2016; Weintrop & Wilensky, 2015).

Block-based programming is a relatively new paradigm compared to the traditional text-based programming and was specifically designed to introduce programming to beginners. Block-based programming environments feature a visual graphic user interface where learners can drag and drop pre-existing code blocks (i.e., built-in functions and statements) and snap them together to create a computer program rather than writing a text-based code (Lye & Koh, 2014; Weintrop & Wilensky, 2015). Figure 1.1 shows a piece of code written in a block-based programming language on the left, and its equivalent of traditional text-based code written in JavaScript programming language on the right. A visual comparison of the two representations readily shows the syntactic simplicity of the block-based code versus the JavaScript code. Blocks in these programming environments generally have unique shapes and different colors based on their functions and are allowed to be snapped only in particular ways depending on their shape. This prevents learner from making
any syntax-related mistakes during the construction of the program. Also, in these environments blocks with similar functions are usually grouped into categories, which makes it easier for learner to browse and find a particular block (Weintrop & Wilensky, 2015). All of these embedded scaffoldings in block-based environments serve to reduce cognitive overload that would result from having to memorize and recall the syntax of programming language and enable learner to focus on constructing algorithms and practicing computational thinking (Lye & Koh, 2014). Some of the well-known environments that utilize block-based programming include Scratch (Resnick et al., 2009), Alice (Cooper, Dann, & Pausch, 2000), StarLogo TNG (Klopfer, Scheintaub, Huang, Wendel, & Roque, 2009), and Blockly (Trower & Gray, 2015).

![Diagram](image)

**Figure 1.1.** Block-based versus traditional text-based programming

The use of visual block-based programming environments in science classrooms can provide students with a scaffolded pathway to learn about computational thinking practices through programming while also engaging in model building activities that can effectively
promote conceptual science learning. A limited number of studies have shown that students from middle grades to high school were able to successfully construct computer models with visual programming environments and improved their understanding of science content (Basu et al., 2012, 2016; Sengupta & Farris, 2012).

Finally, integrating computational thinking practices into science curricula and instruction instead of teaching in only informal, after-school settings is important for addressing the issues related to gender and social equity (Harlow et al., 2016). Although research shows promise for introducing computational thinking through informal learning experiences (e.g., robotics clubs), this approach often results in a population consisting of self-selected male students, or students who are already interested in computers, or those from families with higher socioeconomic status involve in these informal programs (Margolis & Fisher, 2003). The demographic statistics in computer science and other STEM disciplines in the United States show the urgent need for the nation to diversify the STEM pipeline. For example, although women constitute half of the total U.S. college-educated workforce, they represent 29% of individuals in all STEM occupations, and only 25% of computer and mathematical scientists (National Science Board, 2016). Also, several racial and ethnic minority groups continue to be significantly underrepresented, e.g., 70% of workers in STEM workforce are White/Caucasian (National Science Board, 2016). Research has demonstrated the importance of career-related decisions made by children during the middle school years on their future career choices (Tai, Liu, Maltese, & Fan, 2006). Therefore, it is important for all middle school students, not just a certain group, to be
exposed to authentic science practices such as computational thinking and modeling and have favorable attitudes towards STEM fields.

**Theoretical Frameworks**

**Constructionism**

Constructionism is a learning theory developed by Seymour Papert (1980) and builds upon the constructivist ideas of learning. The constructivist approach to learning argues that knowledge is actively constructed by the learner, and learner’s prior knowledge and personal experiences have an important role in the learning process (Piaget, 1965, 1971). Constructivism rejects the idea of learning by transmission of knowledge and suggests that learning is a contextualized process of constructing knowledge rather than acquiring it from a source. Based on his studies on how children understand the world, Piaget (1965) identified mechanisms which include assimilation and accommodation to explain how learner constructs knowledge by building on past experiences. Papert worked with Piaget in Geneva in the late 1950s and later expanded on Piaget's ideas by focusing on the ways of how internal knowledge construction can be supported by real-world objects, particularly computers (Kafai, 2006). Constructionism differs from constructivist framework by emphasizing the importance of externalizations and argues that learning is particularly robust when learners project out their inner ideas by constructing tangible and shareable artifacts (Papert, 1990):

We understand “constructionism” as including, but going beyond what Piaget would call “constructivism.” The word with the ν expresses the theory that knowledge is built by the learner, not supplied by the teacher. The word with the n expresses the
further idea that this happens especially felicitously when the learner is engaged in the construction of something external or at least shareable…a sand castle, a machine, a computer program, a book. This leads us to a model using a cycle of internalization of what is outside, then externalization of what is inside and so on. (Papert, 1990, p. 3)

Similar to Piaget’s constructivist framework, constructionism underlines the "learning by doing" approach, particularly using new technologies to help individuals learn by actively inquiring (Bers, 2010). Constructionist learning theory emphasizes providing learners with opportunities to build external and shareable artifacts, and highlights the personal and social dimensions of knowledge construction (Kafai & Resnick, 1996). According to Piaget (1971), abstract thinking and reasoning is the last stage in the cognitive development of children. However, constructionism suggests that concrete thinking can be as advanced as abstract thinking (Turkle & Papert, 1990). In constructionism, physical or digital objects—as Papert calls "objects-to-think-with"—play a central role for supporting the concrete thinking about abstract phenomena (Kafai, 2006).

Constructionism also places primacy on the situated nature of learning by acknowledging learning as the knowledge construction process in the context of social and cultural participation (Ackermann, 2001; Kafai, 2006). Constructionism is interested in how learners engage in a conversation with their own or other people’s artifacts, and how these conversations boost self-directed learning and ultimately facilitate the construction of new knowledge (Ackermann, 2001). Constructionist theory will guide the design of the learning activities of the intervention where students will construct simulation-based computer models by utilizing a visual programming environment and will use their scientific models to
communicate their ideas in the classroom and further their understanding of science concepts.

**Learning through Constructing Representations**

A large body of research indicates that engaging in constructing external representations of scientific phenomena in the classroom facilitates conceptual understanding of science (Greeno & Hall, 1997; Kozma, 2003; Tytler, Prain, & Peterson, 2007). Different representational systems have different affordances and may emphasize different aspects of a phenomenon. For example, building diagrams can be useful to illustrate the *components* of a system and show how these components are related spatially, while constructing simulation-based computer models can describe the *processes* in the system such as how the components of the system change across time and space (Wilkerson-Jerde, Gravel, & Macrander, 2015). Constructing representations is important in science classroom not only as a product that reflects student understanding and alternative conceptions, but also as a process that furthers student learning (Carolan, Prain, & Waldrip, 2008).

Prain and Tytler (2012) proposed a framework called Representational Construction Affordances (RCA) to explain how and why students learn by constructing representations. By building on previous research (Bransford & Schwartz, 1999; Pickering, 1995), the researchers recognize the idea that by constructing representations “students were productively constrained in their reasoning by having to focus on key aspects of the problem, select appropriate tools, and apply relevant background knowledge to the problem.” (p. 2756). In their framework, Prain and Tytler (2012) identified three interdependent
dimensions which consist of semiotic, epistemic, and epistemological. The researchers argued that “each dimension is linked by its focus on the way representations productively constrain meaning-making practices in science and in science education, taking into account the interplay of diverse cultural and cognitive resources students use to achieve this meaning-making.” (p. 2757). In this framework, semiotic dimension focuses the use of semiotic or cultural tools applicable to constructing representations and developing competence to recognize and use these tools. This is consistent with sociocultural perspectives which place a primacy on the use of cultural tools in mediating learning (Vygotsky, 1981). The epistemic dimension emphasizes the knowledge-building practices such as constructing models, and making, justifying and communicating claims based on the constructed models. Lastly, the epistemological dimension points out that students’ reasoning in science can be improved by the process of constructing and interpreting their own representations.

Prain and Tytler (2012) argued that participating in activities to generate representations in science classrooms leads to high learning gains because building representations: “1) develops students’ cognitive and reasoning strategies relevant to science learning, 2) provides a meaningful, focused context for student development of representational competence, 3) promotes classroom context that engender meaningful communication of science understanding, and 4) is highly engaging for students” (p. 2769). The RCA framework will guide the development of the learning activities of this study by emphasizing the construction of external representations in terms of simulation-based computer models.
Purpose of the Study

The goal of this study was to investigate how attending a one-week intervention course that introduced computational thinking practices and simulation-based model building through a visual block-based programming environment in regular middle school science classrooms influenced seventh-grade students’ understanding of computational thinking concepts and practices as well as their conceptual understanding of a physical science topic, force and motion. The study also examined whether and how seventh-grade students’ attitudes towards programming changed as a result of participating in the intervention course. Since there is a dearth of research on how to integrate computational thinking practices into science curricula and instruction in K-12 education, the findings of this study provided insights on how students’ computational thinking practices and science learning develop by engaging in block-based programming and modeling. Specifically, this study aimed to answer the following research questions:

1. How does building simulation-based computer models of physical phenomena through a visual block-based programming environment in a one-week intervention course influence middle school students’ conceptual understanding of force and motion concepts?

   a. Is there a differential impact of the classroom intervention on students’ conceptual understanding of force and motion concepts according to students’ gender?
2. How does participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment influence middle school students’ understanding of computational thinking concepts and practices?
   a. Is there a differential impact of the classroom intervention on students’ understanding of computational thinking concepts and practices according to students’ gender?

3. How do middle school students’ attitudes towards programming change as a result of participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment?
   a. Is there a differential impact of the classroom intervention on students’ attitudes towards programming according to students’ gender?

Summary

In Chapter One, I described the role of computational thinking as a practice central to all STEM disciplines and highlighted the need to integrate computational thinking practices into K-12 science curricula and instruction. I also explained how simulation-based model building activities can serve as an effective medium to introduce and promote computational thinking concepts and practices, and further conceptual science learning in K-12 science classrooms. Lastly, I introduced the theoretical frameworks and the purpose of the study.
In Chapter Two, the literature pertaining to computational thinking, modeling, and learning science through constructing representations is reviewed. This chapter also includes relevant studies that have investigated the use of computational thinking practices in both formal and informal learning environments.

Chapter Three describes the methodology that was employed for this study. Details on the research design as well as the description of the data sources and analyses are presented.

Chapter Four presents the results of the study that are organized around the research questions.

Chapter Five discusses the results pertaining to each research question and provides explanations of the findings.
CHAPTER TWO: REVIEW OF THE LITERATURE

This chapter presents a review of the relevant literature. First, a historical background of computational thinking given, followed by the sections on definition and practices of computational thinking. Next, the research pertinent to the current state of computational thinking in K-12 education as well as the proposed assessment tools and strategies are discussed.

Since computational thinking was investigated in the context of modeling in this study, a review of the research on the practice of modeling in science and science education was provided, followed by a discussion of studies that investigated modeling as a medium to introduce computational thinking concepts and practices in K-12 science classrooms. Then, the research on learning and reasoning with representations was examined. Lastly, the literature on student learning about force and motion concepts was highlighted.

Historical Background of Computational Thinking

The term “computational thinking” has been introduced by Wing (2006) in her seminal work to refer to a new competency—an analytic ability—that every child should acquire during their formal education in addition to reading, writing, and arithmetic. However, the idea of introducing computation and programming to every student in K-12 education dates back to Seymour Papert’s work at MIT in the 1980s (Grover & Pea, 2013; Barendsen et al., 2015). Papert was involved in the development of Logo programming language which was specifically designed to introduce computation and programming to students of all ages (Papert, 1980). In his book “Mindstorms: Children, Computers, and
Powerful Ideas”, Papert reported his early efforts to teach what he called “procedural thinking” through programming with Logo (Papert, 1980). Procedural thinking included developing, representing, testing, and debugging procedures, which are step-by-step instructions that can be interpreted and carried out by a specified agent such as a computer (Papert, 1980). Logo included a visual output interface in the form of a graphical turtle and a pen attached to it, in addition to a simple and intuitive programming syntax (Guzdial, 2004). Students manipulated the graphical turtle on the computer screen by giving commands in their programs such as “forward 1” (i.e., move one unit forward), “right” (i.e., turn right), and “pendown” (i.e., put the pen down attached to the turtle to draw when the turtle is moving). Papert argued that engaging in programming with Logo helped students think about their own thinking and gain higher-order thinking skills (Papert, 1991).

Papert (1991) claimed that programming is “reflexive” with other domains; i.e., learning programming in the context of other domains (such as mathematics and science) can be easier than learning them separately. Early studies demonstrated how challenging concepts in mostly mathematics and geometry can be addressed in novel ways using Logo programming environment (Guzdial, 2004; Lehrer, Guckenber, & Lee, 1988). For instance, Lehrer et al. (1988) showed that third-grade students who used Logo programming to solve geometry problems performed better in demonstrating their understanding of geometry concepts than those who did not receive any Logo training. The use of Logo programming in educational contexts has continued, and new programming environments were developed based on Logo including NetLogo (Wilensky et al., 2014) and StarLogo (Klopf, Yoon, & Um, 2005), which are still available today.
Another important historical perspective on computational thinking comes from the work of Andrea diSessa in 1990s. Building on his experiences with Logo, diSessa developed another programming environment called Boxer to introduce “computational literacy” in K-12 settings (diSessa et al., 1991; Sherin et al., 1993). diSessa (2000) argued that computers can be a vehicle for a new form of literacy—a digital age competency—that has the potential to provide a universal access to “powerful ideas” and has applications across domains.

Although the idea of computational literacy has similarities with computational thinking (Wing, 2006), diSessa emphasized the use of “computing as a medium” in exploring other domains such as mathematics and science (Grover & Pea, 2013). Boxer was used in classroom settings to teach mathematics and science concepts (Sherin et al., 1993; Sherin, 2001). While the use of Boxer showed potential to support mathematics and science learning, research showed that integrating Boxer programming into teaching was challenging (Sherin et al., 1993).

**Defining Computational Thinking**

There has been an ongoing discussion in the research community about the definition of computational thinking, and its relationship with other types of analytical competencies such as mathematical and engineering thinking (Grover & Pea, 2013; Sneider et al., 2014). Wing (2006) defined computational thinking as a 21st century competency which “involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science” (p. 33). Many researchers found Wing’s description of computational thinking unclear, and proposed their own definitions, which as a result led to a “definitional confusion that has plagued CT as a phrase” (Grover & Pea, 2013,
p. 38). For instance, Denning (2009) defined computational thinking as “thinking with many levels of abstractions, use of mathematics to develop algorithms, and examining how well a solution scales across different sizes of problems” (p. 28). Lu and Fletcher (2009) argued that computational thinking is not about thinking like a computer, but it is about “developing the full set of mental tools necessary to effectively use computing to solve complex human problems” (p. 260).

In order to address the confusion about the definition of computational thinking, National Research Council (NRC) organized two workshops by bringing together leading experts and researchers from a broad range of disciplines in 2010 and 2011. In the first workshop, participants discussed “the nature of computational thinking and its cognitive and educational implications” (NRC, 2010, p. viii). Participants’ definitions of computational thinking fell into a number of categories, from more abstract to fully operationalized. For instance, one participant defined computational thinking “as a way of formulating precise methods of doing things” (p. 11), while another participant characterized it as “the use of computation-related symbol systems (semiotic systems) to articulate explicit knowledge and to objectify tacit knowledge, to manifest such knowledge in concrete computational forms, and to manage the products emerging from such intellectual efforts” (p. 11). What was highlighted in the workshop is the connection between computational thinking and disciplinary knowledge (NRC, 2010). The report suggested that “(1) students can learn thinking strategies such as computational thinking as they study a discipline, (2) teachers and curricula can model these strategies for students, and (3) appropriate guidance can enable students to learn to use these strategies independently” (p. 62). The second workshop, which
was held in 2011, focused on exploring the pedagogical aspects of computational thinking in the context of K-12 education (NRC, 2011). In this workshop, most of the participants agreed on the idea that the power of computational thinking can be best utilized within domain-specific applications. The workshop report highlighted that computational thinking is pervasive in all STEM disciplines, and in order to develop expertise in computational thinking one needs to learn and recognize its applications and uses across domains (NRC, 2011).

According to the recently introduced K-12 Computer Science Framework (2016), computational thinking is “essentially a problem-solving process that involves designing solutions that capitalize on the power of computers” (p. 69). The Framework emphasized that the use of computers in the classroom does not necessarily constitute computational thinking and provided scenarios to distinguish computational thinking from general computer use. For instance, a student who enters data into a spreadsheet and creates a chart does not necessarily use computational thinking. However, this action can require computational thinking if “the student creates algorithms to automate the transformation of the data or to power an interactive data visualization” (p. 70).

As part of a nationwide initiative to introduce computing in K-12 education in the UK, the Royal Society (2012) defined computational thinking as “the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes” (p. 29). Wing (2014) has recently revisited this topic and provided a revised
definition: “computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.” (p. 1).

Computational Thinking Practices

In addition to defining computational thinking, researchers have also worked to identify the key concepts and practices pertaining to computational thinking. While some researchers have focused on recognizing the practices of computational thinking in particular disciplines such as computer science, others have tried to provide a more comprehensive list of these practices applicable across domains. Approaching from a perspective of a problem-solving skill in computer science, Wing (2006) suggested that computational thinking involves thinking recursively, using abstraction and decomposition when solving a complex problem, and judging the solution for both its efficiency and aesthetics. Along the same lines, Denning (2009) proposed that computational thinking involves “thinking with many levels of abstractions, use of mathematics to develop algorithms, and examining how well a solution scales across different sizes of problems” (p. 28). Grover and Pea (2013) have argued that abstraction is the keystone of computational thinking, a characteristic that distinguishes computational thinking from other types of thinking. Other researchers have also underscored the central role of abstraction in computational thinking (Aho, 2012; Barr & Stephenson, 2011; Sengupta et al., 2013).

Csizmadia et al. (2015) defined abstraction as the process of designing a representational artifact of a complex system or process and making it more understandable
by reducing the unnecessary details. Selby (2015) argued that abstraction is the ability to
decide what details of a problem are important, and what details can be ignored when solving
the problem. As the problems become more complex, the use of abstraction becomes
increasingly important in solving those problems (Aho, 2011). According to Wing (2008),
computational thinking involves dealing with abstractions in the following ways: “defining
abstractions, working with multiple layers of abstraction, and understanding the relationships
among the different layers.” (p. 3718).

Abstraction requires generation of a representation that contains only the necessary
information in characterizing the problem or idea (Csizmadia et al., 2015). For example, a
world map showing the political borders of the states is an abstraction because its primary
purpose is to display boundaries of the states, and in doing that it omits several geographical
details such as mountains and other landforms. Similarly, building a physical or computer
model of the solar system to show the relative sizes and distances of the planets to the Sun
would be an example of abstraction in science classroom. In building this model of the solar
system, one can safely ignore the smaller celestial bodies including moons, asteroids, and
comets. Wing (2014) argued that a computer algorithm is an abstraction “that takes inputs,
executes a sequence of steps, and produces outputs to address a desired goal” (p. 2).

Algorithmic thinking—the ability to approach a problem by breaking it into smaller
and solvable parts and formulating the steps to solve them—is another important element in
computational thinking (Denner, Werner, & Ortiz, 2012; NRC, 2010; Voogt et al., 2015;
Wing, 2006). While it originated from the domain of computer science, algorithmic thinking
is a critical mental ability that is applicable in any problem-solving situation. In the context of computation and computer programming, algorithmic thinking involves using the key concepts in computer science such as sequences, conditionals, and loops (Grover, Cooper, & Pea, 2014).

The two important organizations in the United States, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE), have recently worked jointly to form a steering committee and identify computational thinking practices for K-12 education (Barr & Stephenson, 2011). They identified nine essential practices pertaining to computational thinking which include “data collection, data analysis, data representation, problem decomposition, abstraction, algorithms, automation, parallelization, and simulation” (Barr & Stephenson, 2011). The steering committee also provided pedagogical strategies for embedding computational thinking practices across the K-12 subject areas including science, mathematics, social studies, and language arts. For example, the practice of data analysis may be used in statistically analyzing data from an experiment in a science classroom, and in identifying patterns for different sentence types in a language art classroom (Barr & Stephenson, 2011).

Brennan and Resnick (2012) proposed that computational thinking involves three key dimensions: computational concepts (i.e., sequences, loops, parallelism, events, conditionals, operators, data), computational practices (i.e., being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing), and computational perspectives (i.e., expressing, connecting, questioning). Based on their review of relevant
studies, Grover and Pea (2013) suggested that computational thinking encompasses the following list of concepts and practices (p. 40):

- Abstractions and pattern generalizations
- Systematic processing of information
- Symbol systems and representations
- Algorithmic notions of flow of control
- Structured problem decomposition
- Iterative, recursive, and parallel thinking
- Conditional logic
- Efficiency and performance constraints
- Debugging and systematic error detection

It is important to point out that computational thinking is not only about being able to learn programming and create computer programs. Some researchers even suggested that computational thinking practices can be taught without the use of computers (Lu & Fletcher, 2009; Voogt et al., 2015). However, others have argued that programming is not only a fundamental skill in solving computational tasks, but also a necessary tool to demonstrate computational thinking competencies (Grover & Pea, 2013; NRC, 2010). The NRC report on computational thinking described programming as “an essential aspect of computational thinking” (p. 7). Mitchel Resnick—a leading scholar in computer science and one of the participants of the workshop—stated that “computational thinking is more than programming, but only in the same way that language literacy is more than writing. They are both very important.” (NRC, 2010, p. 13). Likewise, Grover and Pea (2013) criticized the efforts like “CS Unplugged” (http://csunplugged.org)—which promotes the idea of
introducing computational thinking without the use of computers—by arguing that these efforts “may be keeping learners from the crucial computational experiences involved in CT’s common practice.” (p. 40).

Computational Thinking in K-12 STEM Education

Despite the growing importance of computational thinking in STEM disciplines, its inclusion into K-12 science curricula and instruction has been very limited in the United States (Grover & Pea, 2013; Voogt et al., 2015). There is a large research base for computer science education in undergraduate settings; however, research on how students learn about computational thinking and computer programming in K-12 settings is under investigated (Voogt et al., 2015). Likewise, little is known about how K-12 students can engage in computational thinking as part of their science learning in formal classroom settings (Grover & Pea, 2013). Existing studies mostly investigated computational thinking in contexts such as game design or storytelling using visual programming environments, or in informal after-school settings through extracurricular activities such as robotics (Bers, 2010; Bers, Flannery, Kazakoff, & Sullivan, 2014; Denner, Werner, & Ortiz, 2012).

Repenning, Webb and Ioannidou (2010) investigated how game development with block-based programming can be used to introduce computational thinking in a one-week course targeted at middle school students. The researchers reported only anecdotal evidence, but the course seemed to have a positive impact on students’ attitudes and engagement towards computational thinking practices: “students who are usually not engaged, are showing strong interest” (p. 269). In another study targeting girls and underrepresented
minorities from 4th to 12th grade, Ericson and McKlin (2012) investigated how an 8-week summer camp program can increase students’ knowledge of computing concepts as well as students’ attitudes towards computing. Students interacted with a variety of computing environments including Lego NXT, Scratch, Alice, and WeDo robotics kits. The researchers administered a pre- and post-test consisted of an 8-item Likert-scale survey to measure students’ attitudes, and a 10-item multiple-choice assessment to measure students’ understanding of computer programming concepts such as loops and conditionals. The results showed that there was a statistically significant increase in both students’ attitudes towards computing and students’ knowledge of programming concepts (Ericson & McKlin, 2012).

In another study with middle school students, Wolz et al. (2011) investigated the integration of computational thinking practices into a journalism course, which was offered in a summer camp followed by an after-school club. The researchers examined the impact of their intervention on students’ attitudes and confidence towards computing and programming. During the intervention, students engaged in writing news stories and developing animations in Scratch visual programming environment to support their stories. The researchers administered surveys, held focus groups, and made observations to assess the effectiveness of the intervention. The results showed that students developed positive attitudes towards computational thinking and programming. At the end of the intervention, students were able to understand that one does not need to be “math type” to do programming and thought that programming is “fun” (Wolz et al., p. 19).
Computational thinking has been introduced to students in elementary and kindergarten schools too (Bers, 2010; Bers et al., 2014; Flannery et al., 2013). For instance, Bers et al. (2014) investigated kindergarten students’ exposure to computational thinking practices in the context of robotics in a classroom intervention study. The goal of the study was “to describe young children’s learning trajectories in computational thinking and capacity to understand computer programming and robotics concepts” (p.147). Students engaged in building robots using commercially available robotics construction kits and used a graphical programming interface to program their robots to do specific tasks. The results showed that the activities were “generally engaging and developmentally appropriate” (p.154), and students were able to learn the key concepts and practices in robotics and computational thinking.

**Assessment of Computational Thinking**

Assessment plays an important role in education. Assessments can provide evidence of student learning and can help teachers make informed decisions in the classroom (Broadfoot & Black, 2004). Without valid and reliable assessment instruments, it is not possible to determine what students learned in a curriculum. If computational thinking needs to make its way into K-12 curriculum, attention must be given to formulating appropriate assessment strategies, and developing valid and reliable assessment instruments measuring computational thinking (Grover & Pea, 2013; Roman-Gonzalez et al., 2016). A number of assessment methods and instruments have been proposed and developed for assessing students’ understanding of computational thinking concepts and practices in recent years.
(Werner, Denner, Campe, & Kawamoto, 2012; Koh, Basawapatna, Bennett, & Repenning, 2010; Dagiene & Sentance, 2016).

Werner et al. (2012) developed a performance assessment called the “Fairy Assessment” to evaluate students’ understanding of abstraction, conditional logic, algorithmic thinking, and other computational thinking concepts in the context of a visual programming environment called Alice. The assessment was designed as open-ended programming tasks in Alice where students were given a storyline and followed the instructions to complete the tasks. A total of 311 middle school students completed the assessment. Students’ work in the form computer programs was graded by two raters using a scoring rubric. The researchers did not conduct a psychometric validation process, but the preliminary results suggested that the assessment was “successful in picking up a range of CT across students, and a variety of types of CT across the three tasks” (p. 218).

Other researchers have proposed and developed similar assessment strategies that are highly context-dependent and not validated using a rigorous psychometric validation process (Basawapatna et al., 2011; Brennan & Resnick, 2012; Moreno-Leon & Robles, 2015). In a recent study, Roman-Gonzalez et al. (2017) developed a general and relatively decontextualized assessment called Computational Thinking Test (CTt) to measure students’ understanding of computational thinking concepts. The assessment consisted of 28 multiple-choice questions and was administered to a total of 1251 students from 5th to 10th grade, and the validity and reliability of the test was assessed using standard psychometric methods. The
results indicated that the test can be used for collecting quantitative data in pre- and post-
assessment of student knowledge of computational thinking practices.

**Models and Modeling in Science**

Models and modeling play a central role in the conduct of scientific research (Gilbert, 1991; Schwarz & White, 2005). Scientists spend a great deal of their time in building,
testing, studying and revising models, often using computational tools. Models are used
ubiquitously in the physical and biological sciences to describe natural phenomena and
explain causal relationships (Bailer-Jones, 2003). Some examples of well-known models in
science include the double helix model of DNA, the Bohr model of the atom, and the Lotka-
Volterra model of predator-prey interaction. Advancements in computing and technology
have further emphasized the centrality of modeling in science by making model construction
more accessible to scientists today (Schwarz & White, 2005).

A scientific model can be defined as a simplified representation of a natural
phenomenon, an object, or an idea (Gilbert, Boulter, & Elmer, 2000; Schwarz et al., 2009).
For example, a physical scale model of the solar system represents the relative sizes of the
planets in the solar system, while the Big Bang model represents the idea of how the universe
began fourteen billion years ago. While models refer to representational products, modeling
can be described as the epistemic process of building, modifying, evaluating, and reasoning
with models (Lehrer & Schauble, 2006). Models are by their nature imperfect which means
they represent the target phenomena to only a certain degree. They are built to serve a
specific purpose, contain only the most relevant information about the target phenomenon
(Gilbert & Watt-Ireton, 2003). Bailer-Jones (2003) described the models in science as “purely heuristic devices” (p. 60). Models can be used to make abstract scientific entities and concepts visible (Francoeur, 1997), and provide descriptions as well as predictions of complex systems (Gilbert, Boulter, & Rutherford, 1998). Models can allow scientists to carry out experiments that would otherwise be too long, expensive or dangerous to investigate (NRC, 2011b).

Models and Modeling in Science Education

Since modeling is a key practice in science, researchers and policymakers have long advocated the inclusion of modeling into K-12 science curricula and instruction (Harrison & Treagust, 1996; Clement, 2000; Gilbert, Reiner, & Nakhleh, 2008; NRC, 1996). It is important that students should learn about both the ideas and the practices of science. National science education standards have consistently included modeling as an essential practice to engage for all students in K-12 science classrooms (AAAS, 1994; NRC, 1996, 2007; NGSS Lead States, 2013). A Framework for K-12 Science Education report prepared by the NRC identified “developing and using models” as one of the eight key science and engineering practices in K-12 science education (NRC, 2012). The recent introduction of the NGSS, which was based on the NRC’s Framework, emphasized a new vision of integrating these practices with disciplinary core ideas in science to provide students with a meaningful learning context in the classroom (NGSS Lead States, 2013).

A large body of research reported positive effects of model-based pedagogies on science learning across different subject areas and grade levels (Gobert & Clement, 1999;
Khan, 2007; Schwarz & Gwekwerere, 2007; Wu, Krajcik, & Soloway, 2001). For instance, Wu et al. (2001) investigated how high school students developed their understanding of chemical representations as a result of participating in a six-week classroom intervention where students built both physical and 3-D computer models of molecular compounds using physical model kits and a visualization software called eChem. The researchers collected multiple sources of data including classroom video recordings, pre- and post-tests, modeling artifacts, and student interviews. The results showed that students’ conceptual understanding of chemical representations and concepts improved significantly. The results also suggested that “computerized models can serve as a vehicle for students to generate mental images” (p. 821).

Enabling students to engage in modeling practices, which include constructing, evaluating, and revising models, can promote conceptual as well as epistemological understanding of science in the classroom (Schwarz et al., 2009). However, teaching about and learning with models may pose some challenges in the classroom. For example, students may not interpret the models accurately, or find it difficult to understand multiple and competing models of the same scientific phenomenon (Harrison & Treagust, 2010). In addition, teachers may not have the necessary knowledge and skills to be able to construct and use models as well as the pedagogical content knowledge to facilitate model building activities in the classroom settings (Justi & Gilbert, 2002). In their learning progression for scientific modeling, Schwarz et al. (2009) emphasized the importance of metamodeling knowledge—“knowing the forms and purposes of models and criteria for evaluating them” (p. 635)—when students engage in modeling activities in the classroom.
Computational Thinking and Modeling

K-12 Computer Science Standards, which was developed jointly by the Computer Science Teachers Association (CSTA) and the Association for Computing Machinery (ACM), recommends integrating computational thinking with curricular domains such as science and mathematics, rather than teaching it as a separate topic in primary and secondary schools (Tucker et al., 2011). In the context of K-12 science education, the development of students’ computational thinking practices can be synergistically supported by scientific modeling in the classroom (Basu et al., 2016; Sengupta et al., 2013). Wilensky et al. (2014) argued that integrating computational thinking practices into science courses in the context of modeling can help expose all students in a school to computational thinking practices as well as promote a more authentic science learning.

There is limited yet growing research on investigating the computational thinking practices in science classrooms in the context of scientific modeling (Basu et al., 2012; Sengupta, Farris, & Wright, 2012; Sengupta et al., 2013; Wilensky et al., 2014). For instance, Basu et al. (2012) worked with sixth-grade students (N = 24) in a public middle school in the southeastern U.S. to investigate whether engaging in modeling through a customized visual programming environment helped students improve their understanding of kinematics and ecology concepts. Students were divided into two groups where one group (n = 15) received direct one-on-one scaffolding while the other group (n = 9) received only partial scaffolding in the classroom. Pre- and post-assessment of science knowledge showed statistically significant learning gains for both kinematics and ecology units for one-on-one scaffolding group, while the partially scaffolding group demonstrated significant learning gains for only
ecology unit (Basu et al., 2012). The researchers did not investigate how the intervention influenced students’ understanding of computational thinking concepts and practices.

In another study targeting middle and high school students, Sengupta et al. (2012) designed a series of learning activities, which included constructing simulation-based scientific models with a visual programming environment called ViMAP, and investigated how engaging in these activities supported students’ learning of kinematics concepts. The researchers were interested in “integrating programming and computational modeling seamlessly with classroom physics instruction in the particular domain of kinematics” (p. 24). In the beginning of the activities, students learned about basic programming concepts such as loops, conditionals, and debugging in ViMAP environment. Then, they engaged in creating models of physical phenomena relating to kinematics such as objects rolling down on inclined planes, or being pushed on a horizontal surface. The results based on observational data indicated that students need both teacher-led and software-embedded scaffolds when they engage in modeling activities using ViMAP in the classroom. The researchers also suggested that the activities helped students “develop new meanings and ways of generating inscriptions, and use them to represent kinematic phenomena in non-canonical ways” (p. 40).

**Learning and Reasoning with Representations**

Previous research has demonstrated that different modes of representations provide different learning opportunities for students, especially when learning about complex ideas (Ainsworth, 2006; Schnotz & Bannert, 2003). A large body of research shows that the
practice of constructing external representations (e.g., written, verbal, visual) can significantly facilitate conceptual science learning (Prain & Tytler, 2012; Schnozt & Bannert, 2003; Tytler, Prain, Hubber, & Waldrip, 2013). Generating representations is valuable both as a process and a product because it makes one’s mental structures visible and explicit (Ainsworth, Prain & Tytler, 2011). Students learn science better when they engage in the design and consequential use of external representations in the classroom (Kolodner et al., 2003). Representations can be either dynamic (e.g., animation, simulation) or static (e.g., plots, diagrams). Dynamic representations have distinct advantages over static representations (Ainsworth & Van Labeke, 2004). In particular, generating external representations in terms of simulation-based dynamic computer models can be a powerful strategy to facilitate conceptual learning (Basu et al., 2016; Sengupta et al., 2013; Van Labeke & Ainsworth, 2002).

Representations in science are built using a wide range of semiotic tools and resources, and can provide semantically rich information, which can be utilized for scientific reasoning and problem solving (Giere, Bickle, & Mauldin, 2006). While representations are products of scientific inquiry, the construction and use of representations refer to the epistemic processes that promote the development of new understanding (Carolan, Prain, & Waldrip, 2008; Kozma, 2003). From a constructionist perspective (Papert, 1993), there is an important distinction between constructing one’s own external representation (e.g., drawing the circulatory system of the human body on a science notebook) versus using a prefabricated one (e.g., a textbook diagram the circulatory system of the human body). An extensive body of literature has demonstrated that constructing, evaluating, and reasoning with
representations is crucial for conceptual science learning (Ainsworth, 2008; Greeno & Hall, 1997; Hubber, Tytler, & Haslam, 2010). Cox (1999) argued that the effectiveness of a particular representation in a specific context depends on a 3-way interaction between “(a) subject factors (e.g. prior knowledge, cognitive style), (b) the semantic properties of the representation, and (c) the demands of the task” (p. 360).

From a sociocultural perspective, external representations play an important role in the learning process (Vygotsky, 1980; Ford & Forman, 2006). Representations can be constructed for exclusively personal use, for communicating ideas and information with others, or both. In classroom settings, learners construct representations usually for both personal and social use. Students often share their representations with others (i.e., teachers and students) in the classroom. They compare, comment and critique each other’s representations. This process can greatly facilitate epistemic knowledge construction in science classrooms by enabling students to externalize their mental structures, comparing and contrasting them with others’, and adjusting and modifying their understanding in the process (Tytler & Hubber, 2016).

Learning about Force and Motion Concepts

Force is one of the core concepts in physics, which is used to describe the behavior of objects in electromagnetic and gravitational fields. All forces arise from interactions between any two objects, which in result can cause changes in one or both of them. A conceptual understanding of force is important for describing how interaction between objects leads to a change in their motion (NRC, 2012). At the macro scale level, the motion of an object is
governed by the Newton’s second law of motion which states that the net force acting on an object is directly proportional to its acceleration (net force = mass x acceleration).

Students’ conceptual understanding of force and motion concepts has been extensively studied in the literature (Alonzo & Steedle, 2009; Eryilmaz, 2002; Hestenes, Wells, & Swackhamer, 1992; Kozhevnikov & Thornton, 2006; Reiner, Slotta, Chi, & Resnick, 2000; Yuruk, Beeth, & Andersen, 2009). Previous research repeatedly showed that students from primary school to college hold several misconceptions about force and motion concepts, which are quite robust and persistent (Eryilmaz, 2002; Neumann, Fulmer, & Liang, 2013). For example, students tend to think that there should be a force acting on an object if the object is moving (Halloun & Hestenes, 1985), or that force is property of a single object rather than a feature of the interaction between two objects (Reiner et al., 2000). Some students confound acceleration with velocity, thinking that when two objects have the same velocity they should have the same acceleration (Eryilmaz, 2002). Students often have difficulty conceptualizing a frictionless environment, and think that a continuously applied force is necessary for the motion of an object at constant velocity even on a frictionless medium (Halloun & Hestenes, 1985). Some students think that velocity of an object is directly proportional to the applied force, therefore constant force means constant velocity (Sequeira & Leite, 1991). Another common misconception among students is that they believe heavier objects fall faster in free fall, which is related to students’ difficulty visualizing a frictionless medium (Eryilmaz, 2002). Besides, students’ interpretation and use of the scientific terms may not correspond to their actual scientific meaning. For instance, in their study with primary and middle school students, Ioannides and Vosniadou (2001)
identified four well-defined and internally consistent interpretation of the term force, and none of these interpretations aligned with its actual scientific meaning. The researchers also found that students’ understanding of the term force varied by age, with younger students considered it as an internal property of all objects while older students interpreted the term force as “an internal property of objects that move, as the result of an agent pushing or pulling them” (p. 2).

It is important to note from a theoretical perspective that some researchers identified students’ prior knowledge and preconceptions that are not consistent with expert understanding as “intuitive” and “useful” cognitive resources (diSessa, 1993) instead of referring them as misconceptions or alternative conceptions. In this model, students’ intuitive knowledge (i.e., prior knowledge) is made up of smaller and fragmentary cognitive structures, which diSessa called phenomenological primitives, or p-prims for short (diSessa, 1988; 1993). diSessa (1993) argued that students use or “activate” different p-prims to interpret their observations about physical phenomena. For example, when students are asked why they think it is hotter during the summer compared to the winter, they may respond that it is because the Earth is closer to the Sun during the summer (Hammer, 1996). diSessa (1993) argued that in this example students are activating a particular p-prim called “closer means stronger”. This idea that closer means stronger is not totally wrong because it can be used to explain other physical phenomena such as it gets hotter and brighter when you get closer to a bonfire or sound gets louder when you get closer to the speaker. diSessa (1993) argued that the p-prim “closer means stronger” is not incorrect but its activation to explain certain situations is incorrect. This perspective has some implications for classroom
instruction. Instead of trying to eliminate the p-prim, teacher can help students figure out when to use certain p-prims to explain physical phenomena.

**Summary**

This chapter provided a review of the literature relevant to computational thinking and modeling in K-12 education. Also, research on learning and reasoning with external representations as well as student understanding about force and motion concepts were presented. The next chapter describes the methodology of the study including the research design, data sources, and data analysis procedures.
CHAPTER THREE: METHODOLOGY

The purpose of this study was to investigate how participating in a week-long classroom intervention that was designed to introduce computational thinking practices in the context of scientific model building through a visual block-based programming environment influenced middle school students’ understanding of computational thinking practices as well as conceptual understanding of force and motion concepts. The study also examined how middle school students’ attitudes towards programming changed as a result of participating in the intervention course. A convergent mixed methods research design (Creswell, 2015) was used to collect both quantitative (i.e., pre- and post-test) and qualitative data (i.e., interviews with students, reflection statement responses, classroom observation notes) to address the following research questions of this study:

1. How does building simulation-based computer models of physical phenomena through a visual block-based programming environment in a one-week intervention course influence middle school students’ conceptual understanding of force and motion concepts?

   a. Is there a differential impact of the classroom intervention on students’ conceptual understanding of force and motion concepts according to students’ gender?

2. How does participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment influence middle school students’ understanding of
computational thinking concepts and practices?

a. Is there a differential impact of the classroom intervention on students’ understanding of computational thinking concepts and practices according to students’ gender?

3. How do middle school students’ attitudes towards programming change as a result of participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment?

a. Is there a differential impact of the classroom intervention on students’ attitudes towards programming according to students’ gender?

This chapter begins with a discussion of the research design, followed by a description of the research setting and participants. Next, the details about the intervention, data sources including the validity and reliability of the data collection instruments, and data analysis procedures are given.

**Research Design**

This study employed a convergent mixed methods research design, using a combination of quantitative and qualitative research approaches (Creswell, 2015). The mixed methods design was chosen to provide a richer and deeper understanding of how middle school students’ understanding of computational thinking practices and force and motion concepts as well as their attitudes towards programming change by engaging in programming and simulation-based model building activities through a visual block-based programming
environment. Creswell and Clark (2011) defined mixed methods research design as a procedure for collecting, analyzing, and mixing both quantitative and qualitative research techniques in a single study or a series of studies to address a research problem. The basic intention for the use of quantitative and qualitative approaches in combination is to provide a more complete understanding of the research problem than either method alone (Teddle & Tashakkori, 2009).

A convergent design was chosen for this mixed methods study (Creswell, 2015). The motivation for the use of convergent design in mixed methods research is that it combines the strengths of both quantitative and qualitative approaches: “quantitative data provide for generalizability, whereas qualitative data offer information about the context or setting” (Creswell, 2012, p. 542). In convergent design, the researcher collects both qualitative and quantitative data separately during the course of the study with the intent to merge the results of the quantitative and qualitative data analyses to address a research problem as “both forms of data provide different insight, their contribution to seeing the problem from multiple angles and multiple perspectives” (Creswell, 2015, p. 36). After the results are merged from the two datasets, researcher examines to what degree the quantitative results are confirmed by the qualitative findings, or vice versa (Creswell, 2015). Creswell and Clark (2011) suggested that triangulation of the quantitative and qualitative data analysis provides greater validity through "mutually corroborated" findings (p. 62). In this study, each research question was answered by collecting both quantitative and qualitative data relevant to the research question, analyzing the datasets, and merging the results with the intention that both forms of data will provide unique insights.
There are several approaches to merge and compare the results of both quantitative and qualitative data analysis in a convergent mixed methods design (Creswell, 2012). A popular approach is to provide a description of both qualitative and quantitative results next to each other, and interpret whether and how qualitative findings align with the statistical results, or vice versa (Creswell, 2012). Another approach includes transforming one of either dataset, so that qualitative and quantitative datasets can be directly compared with each other (Creswell, 2012). For example, qualitative themes that emerge from interviews can be quantified by giving a score for their frequency, and then the quantitative measures assessing the same construct or idea can be compared with these scores (Creswell, 2012). In this study, the first approach was used; the researcher provided the results of quantitative and qualitative data analyses together, and interpreted how they—individually and in combination—address the research questions.

Figure 3.1 shows the research design of this study, illustrating the flow of activities and procedures at each stage. The researcher spent seven days in total in the classroom, one day for the pre-test, five days for the classroom intervention, and one day for the post-test. Before the classroom intervention, students were asked to complete a pre-test which consisted of two multiple-choice assessments and a short attitudinal survey (Phase 1). The pre-test took about 50 minutes for students to finish and was completed on Day 1. On Day 2, the researcher began conducting the intervention in the classroom that continued for five days (i.e., five class periods in total for each class), introducing computational thinking practices and simulation-based model building through block-based programming (Phase 2). During the course of the intervention, two forms of qualitative data including students’
written responses to reflection statement questions at the end of each class period and
detailed classroom observation notes were collected. The intervention was completed on Day
6. Students completed the post-test (Phase 3), which was identical to the pre-test, on Day 7.
During the next two weeks after the post-test, the researcher conducted structured interviews
with a subset of students (n = 12), which took place in the school library (Phase 4). After the
data collection was completed, the researcher started working on data analysis using both
quantitative and qualitative methods, and generated the results (Phase 5).

Figure 3.1. Convergent mixed methods design of the study

Research Setting

This study took place in a public middle school in the southeastern U.S. during the
spring semester of 2017. The school was recently designated by the school district as a
“magnet school for the digital sciences” and offers elective classes on basic computer science
topics. The school was selected for this study because of the researcher’s familiarity with the
school and school district. The science teacher and the school magnet coordinator were also
interested and willing to participate in the study, making it a convenience sample (Gall, Gall,
& Borg, 2006). The intervention was conducted by the researcher himself in one of the computer labs in the school during the regular seventh-grade science class periods. One class period was about 50 minutes.

**Participants**

A seventh-grade science teacher who was teaching five different classes was recruited to participate in this study. The total number of students in the teacher’s classrooms was 132. After receiving an IRB approval, the researcher visited the recruited teacher’s classrooms. After introducing himself and explaining the purpose of the study, the researcher invited all students to participate in the study. The researcher distributed to the students the hard copies of the Parent Consent Form (Appendix A), and Student Assent Form (Appendix B), and requested written consent from parents as well as written assent from students who were interested and wanted to participate in this study. The researcher strongly emphasized that participation in the study is voluntary, and participation or nonparticipation to the study will not affect students’ school grades in any way. In addition, the researcher stated that students will be doing the same learning activities in the classroom during the intervention regardless of their participation to the study, but only participating students’ data will be collected for research purposes.

Eighty-two seventh-grade students in total consented to participate in the study. 54% of the participating students were male (n = 44), and 46% were female (n = 38). The age of the students ranged from 12 to 14, with an average of 12.7 (SD = 0.52). The classrooms were ethnically and racially diverse, with 20% of the students were African-American, 21% were
Latino-Hispanic, 5% were Asian-American, 4% were Indian-American, and the rest of the students (50%) was non-Hispanic White. While no data was available nor was collected regarding participating students’ socioeconomic backgrounds, about 53% of students in the school received free or reduced lunch.

**Intervention**

The purpose of the classroom intervention designed for this study was to introduce seventh-grade students to computational thinking concepts and practices in the context of building simulation-based models of physical phenomena through a visual block-based programming environment in formal classroom settings. The intervention aimed to further students’ knowledge of computational thinking practices as well as conceptual understanding of force and motion concepts. The intervention was carefully designed by the researcher using his past experience as a physics teacher as well as his experience in teaching introductory programming when he was running an after-school robotics club. Scratch (https://scratch.mit.edu) was chosen as the visual block-based programming environment that students worked with during the course of the classroom intervention. Scratch was developed at the Massachusetts Institute of Technology (MIT) as a freely available open source software to teach computer programming through a visual block-based environment (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). Scratch can work on a browser with an internet connection or can be downloaded and installed on a computer to work offline. Figure 3.2 shows the user interface of Scratch with a sample piece of code.
Scratch was chosen as the programming environment for the intervention of this study for several reasons. First, several studies identified ways Scratch can be used as an effective tool to build simulation-based models for middle and high school students (Balouktis, 2016; Lopez & Hernandez, 2015). Second, Scratch can work both online and offline, which makes it suitable for classroom settings where the internet connection is often not reliable. Lastly, Scratch is one of the most popular visual programming environments today with over 17 million registered users (Scratch Statistics, 2017), and as a result provides extensive documentation for both students and educators on its website (Scratch Wiki, 2017).

Figure 3.2. User interface of the Scratch programming environment
The classroom intervention took five days (i.e., one class period at each day) to complete. During the first three days of the intervention, students learned about basic computational thinking practices (e.g., abstraction, algorithms) as well as the fundamental concepts in computer programming (e.g., variables, conditionals, loops) using the Scratch programming environment. In the next two days, students engaged in simulation-based model building activities in Scratch to demonstrate and further their conceptual understanding of force and motion concepts. It is important to note here that students studied force and motion unit in the previous semester (i.e., Fall, 2016), so the purpose of the modeling activities was to address any misconceptions students had and further improve their conceptual understanding of force and motion concepts. Table 3.1 shows the contents of the learning activities in the classroom for each day. In addition, Table 3.2 presents the state science standards (NC Essential Science Standards, 2011) that were addressed during the modeling activities.
Table 3.1  

*Computational thinking practices and modeling activities covered during the intervention*

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>Introduction to block-based programming and Scratch user interface; problem decomposition and creating algorithms as a set of instructions; using abstractions; conditionals and decision-making in programming (if and if-else statements); using mathematical operators in programming.</td>
</tr>
<tr>
<td>Day 2</td>
<td>Loops and different repetition structures in programming; using nesting in programming; using logical operators in programming.</td>
</tr>
<tr>
<td>Day 3</td>
<td>Creating, using and manipulating different types of variables in programming; event-driven programming with broadcasting in Scratch.</td>
</tr>
<tr>
<td>Day 4</td>
<td>Modeling building activity 1 – Building a model that will simulate the motion of a car on a frictionless road with input variables of force applied on the car and mass of the car, and with output variables of velocity and acceleration of the car.</td>
</tr>
<tr>
<td>Day 5</td>
<td>Modeling building activity 2 – Building a model that will simulate the fall of a basketball in frictionless medium (i.e., free fall) with input variables of gravitational acceleration and mass of the basketball, and with output variables of velocity and weight of the basketball.</td>
</tr>
</tbody>
</table>

Table 3.2  

*State Science Standards addressed during the modeling activities*

<table>
<thead>
<tr>
<th>Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.P.1</td>
<td>Understand motion, the effects of forces on motion and the graphical representations of motion.</td>
</tr>
<tr>
<td>7.P.1.1</td>
<td>Explain how the motion of an object can be described by its position, direction of motion, and speed with respect to some other object.</td>
</tr>
<tr>
<td>7.P.1.2</td>
<td>Explain the effects of balanced and unbalanced forces acting on an object (including friction, gravity and magnets).</td>
</tr>
<tr>
<td>7.P.1.3</td>
<td>Illustrate the motion of an object using a graph to show a change in position over a period of time.</td>
</tr>
</tbody>
</table>
Figure 3.3 shows the screenshot of the first modeling activity that students engaged in on Day 4. In this activity, students were asked to build a model to simulate the motion of a car on a frictionless road with a given force and mass of the car. Students were provided with the sprite (i.e., the car) and the stage (i.e., the road) at the beginning of the activity to save time, and they were asked to develop an algorithm and construct the script for this model. Students were told that this model should have at least four variables: one for manipulating the force applied on the car, one for manipulating the mass of the car, one for showing the instantaneous velocity of the car in real time, and one for showing the instantaneous acceleration of the car in real time. After students finished building the model, they were asked to experiment with the model by running it with different parameters (i.e., changing the input variables of force and mass). A number of students were also asked to come to the whiteboard and explain how the algorithm works to the whole class by going through the code step by step.
Figure 3.3. Screenshot of the first modeling activity in Scratch

Figure 3.4 shows the screenshot of the second modeling activity that students engaged in on Day 5. In this activity, students were asked to build a model to simulate the fall of a basketball under the sole influence of gravitational force (i.e., free fall). Students were provided with the sprite (i.e., the basketball) and the stage (i.e., the picture with a ground represented by a black line at the bottom), and they were asked to develop an algorithm and construct the script for this model, just like they did in the previous day. Students were told that this model should have at least four variables: one for manipulating the gravitational acceleration, one for manipulating the mass of the basketball, one for showing the instantaneous velocity of the basketball in real time, and one for showing the weight of the basketball in real time. After students finished building the model, they were asked to experiment with the model by running it with different parameters (i.e., changing
the input variables of gravitational acceleration and mass). Similar to the previous modeling activity, a number of students were again asked to come to the whiteboard and explain how the algorithm works to the whole class by going through the code step by step.

![Activity-2](image)

*Figure 3.4. Screenshot of the second modeling activity in Scratch*

During the learning activities, students worked individually. However, they were encouraged to talk and ask questions to each other when they needed help since the Constructionist approach to learning emphasizes the social dimension of knowledge construction (Kafai & Resnick, 1996). Students who did not provide consent to participate in the study still took part in the learning activities during the intervention. However, no data was collected from non-consenting students.
Data Sources

Data for this study were collected from a variety of sources including two multiple-choice assessments, a Likert-style attitudinal survey, students’ written responses to reflective statement questions, detailed classroom observation notes, and structured interviews with students. Below is a description of the sources of data collected for this study.

Pre-test and Post-test

The pre- and post-test consisted of a 28-item multiple-choice assessment that aimed to assess students’ understanding of computational thinking concepts and practices (Roman-Gonzalez et al., 2017), a 16-item multiple-choice assessment that aimed to assess students’ conceptual understanding of force and motion concepts (Alonzo & Steedle, 2009), and an 8-item Likert-style survey on a 5-point scale (1: Strongly Disagree, 5: Strongly Agree) that aimed to assess students’ attitudes towards programming. The pre- and post-test were identical. The pre- and post-test were completed by students during the same time period of the study, one day before the intervention took place and one day after the intervention was over. The pre- and post-test were taken online using the Qualtrics website. The pre- and post-test data served as the primary quantitative data that were collected in this study.

The 28-item multiple-choice assessment in the first part of the pre/post test, which aimed to assess students’ understanding of computational thinking concepts and practices, was called the Computational Thinking Test (CTt) and was developed by Roman-Gonzalez et al. (2017). The validity and reliability of the CTt were assessed by administering the test to a large sample of students (N = 1251). The initial version of the CTt consisted of 40 multiple-
choice questions. After going through a content validation process with a panel of experts, a final version of the assessment was constructed, which consisted of 28 multiple-choice questions with four answer options (only one correct). Figure 3.5 shows a typical question type from the CTt, which was adapted from Roman-Gonzalez and colleagues (2017). The final version of the CTt was administered to 1251 students of age 10 to 16 (i.e., 5th grade to 10th grade). Fifty-eight percent of the students (n = 730) were male. The results showed that students’ scores were almost normally distributed, and the overall test had an appropriate item difficulty of $p = 0.59$ (medium), ranging from $p = 0.16$ (item 23) to $p = 0.96$ (item 1).

The reliability of the test was measured by Cronbach’s alpha (Cronbach, 1951) and found to be $\alpha = 0.793$, indicating good internal consistency. The criterion validity of the test was assessed with respect to two standardized psychological tests: The Primary Mental Abilities (PMA) battery (Hertzog & Bleckley, 2001), and the RP30 problem-solving test (Seisdedos, 2002). The PMA battery intended to measure several cognitive abilities including verbal, spatial, numerical, and reasoning. Statistically significant positive correlations were found between the CTt and spatial ability ($r = .439, p < .01$), reasoning ability ($r = .442, p < .01$), verbal ability ($r = .273, p < .01$). Similarly, a statistically significant positive correlation was found between the CTt and RP30 problem-solving test ($r = .669, p < .01$).
Figure 3.5. A typical question from the CTt (adapted from Roman-Gonzalez et al., 2017)

The second part of the pre/post test consisted of 16-item multiple-choice assessment that was developed by Alonzo and Steedle (2009) to assess middle school students’ conceptual understanding of force and motion concepts. The 16-item assessment was developed based on the Force and Motion Learning Progression (LP) that was developed by the same researchers (Alonzo & Steedle, 2009). The LP outlines students’ conceptions about the basic concepts relating to force and motion, including “the different conditions under which an object is in motion and the conditions under which an object is at rest” (p. 396). The LP consisted of five levels, ranging from no evidence of student understanding of force and motion concepts (level zero) to an expert level of understanding (level five). Alonzo and Steedle (2009) developed the multiple-choice items in a way that item options reflect different levels of understanding in the LP. This type of questions is also called “ordered multiple-choice” (OMC) questions (Alonzo & Steedle, 2009). In OMC questions, “each
response option corresponds to a level of the learning progression so that a student’s responses to a set of items could be used to characterize his or her understanding of the target concept” (Alonzo & Steedle, 2009, p. 390). This means that while the correct option links to the target level of understanding in the LP, other options are associated with lower levels of understanding in the LP and can still provide useful information about students’ (mis)conceptions of the relevant science concepts. Table 3.3 shows a sample question from the Force and Motion Assessment. Alonzo and Steedle (2009) administered the multiple-choice assessment to small group of students \(N = 64\), and the reliability of the test was assessed by Cronbach's alpha. The results showed a moderate reliability for the test \(\alpha = .523\). In a recent study, Kizil (2015) administered the test to a large sample of students \(N = 931\), and the results also suggested moderate reliability for the test \(\alpha = .53\).

Table 3.3

A sample question from the Force and Motion Assessment (Alonzo & Steedle, 2009)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>There is an unbalanced force acting on the spacecraft.</td>
</tr>
<tr>
<td>B.</td>
<td>The spacecraft must have an engine which is exerting a constant force on it.</td>
</tr>
<tr>
<td>C.</td>
<td>There are no forces acting on the spacecraft.</td>
</tr>
<tr>
<td>D.</td>
<td>The force that launched the spacecraft into outer space is still acting on it.</td>
</tr>
</tbody>
</table>

The last part of the pre/post test was a short survey (Appendix C) that aimed to measure students’ attitudes towards programming. The survey consists of 8 Likert-style items on a five-point agreement scale (1: Strongly Disagree, 5: Strongly Agree), which were adapted from the science subscale of “the Middle/High School Student Attitudes towards
STEM” (S-STEM) Survey developed by Unfried, Faber, Stanhope, and Wiebe (2015). The S-STEM Survey consisted of six sections: four sections measuring student attitudes towards science, mathematics, engineering, and the 21st century skills; one section measuring student interest in STEM careers, and one section collecting background information about student. Self-efficacy and outcome expectancy were the main psychological constructs underlying the attitudinal survey items (Unfried et al., 2015). The content validity of the survey was assessed by a panel of three subject matter experts. Some items in the survey were revised in this process such as changing some negatively worded phrases to positive ones to reduce potential confusion (Unfried et al., 2015). The survey was administered to a large sample of sixth- through twelfth-grade students ($N = 9081$) in the fall and winter of 2011 and 2012. The researchers used exploratory and confirmatory factor analysis to assess the construct validity of the survey. Factor analyses were conducted on individual scales as well as on all attitude questions as a whole. The results of the factor analyses showed a clear factor structure which nearly all items loaded significantly on their expected constructs. The results also showed that no item did cross-load. The reliability of the survey was calculated by Cronbach’s alpha and found to be $\alpha = 0.83$, indicating high internal consistency. The science scale of the survey consisted of nine items. These items were modified, with the key word “science” in the items replaced with “programming”. One item that reads “I can handle most subjects well, but I cannot do a good job with science” was dropped because it was the only item with negative wording that may lead to confusion.
Reflective Statement Questions

During the intervention, students who consented to participate in the research responded online to the reflective statement question of “What did I learn today?” at the end of each class period before leaving the classroom, using the Qualtrics website. Students were told that their responses were research purposes only and had no effect on students’ class grades. On average, students spent three to four minutes responding to the reflective statement question at the end of each class period.

Interviews with Students

During the last day of intervention, the researcher recruited twelve students (at least two students from each class section) by providing information about the purpose and the format of the interview using the “Volunteering for Interview” script (Appendix D). Participation to the interview was completely voluntary, and students who volunteered for the interview received a $10 Target gift card as a compensation for their time. Only students who consented to participate in the study were allowed to volunteer for the interview. At the end of Day 5, the researcher collected students’ names who wanted to participate in the interviews, and then planned the interview dates with them.

For the next two weeks after the completion of the classroom intervention, the researcher conducted structured interviews with twelve volunteering students. Seven of the volunteering students were male, and five of them were female. The sample was also ethnically diverse: seven of them were non-Hispanic White, three were African-American, and two were Latino-Hispanic. During the interviews, the researcher asked questions about
the computer models students had built, using the Interview Protocol (Appendix E) to further explore their conceptual understanding of force and motion concepts as well as their understanding of computational thinking concepts and practices introduced in the intervention. Interviews took place in the school library during the lunch breaks and lasted about 30 minutes. The researcher audio-recorded the interviews and transcribed them verbatim for qualitative data analysis.

**Classroom Observations**

While the researcher was conducting the intervention, another doctoral student was present in the classroom and took extensive classroom observation notes throughout the intervention. These notes helped assess the fidelity of the intervention, i.e., described and documented how the learning activities went in the classroom, whether anything was too difficult or too simple for students, what kind of challenges students faced during the activities. The classroom observation notes also helped the researcher understand students’ engagement in the learning activities.
Table 3.4
The list and descriptions of the data sources

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational Thinking Test</td>
<td>Quantitative</td>
<td>28 multiple-choice questions (Roman-Gonzalez et al., 2016)</td>
</tr>
<tr>
<td>Force and Motion Assessment</td>
<td>Quantitative</td>
<td>16 multiple-choice questions (Alonzo &amp; Steedle, 2009)</td>
</tr>
<tr>
<td>Attitudinal Survey</td>
<td>Quantitative</td>
<td>8-item Likert-style survey on a 5-point scale (adapted from Unfried et al., 2015)</td>
</tr>
<tr>
<td>Reflective Statement Questions</td>
<td>Qualitative</td>
<td>Students’ written responses to the reflective statement question of “What did I learn today?” at the end of each class period</td>
</tr>
<tr>
<td>Student Interviews</td>
<td>Qualitative</td>
<td>Transcriptions of the audio records of the structured interviews with a subset of the participating students after the intervention</td>
</tr>
<tr>
<td>Classroom Observation Notes</td>
<td>Qualitative</td>
<td>Extensive classroom observation notes taken by a doctoral student who was always present in the classroom during the intervention</td>
</tr>
</tbody>
</table>

Data Analysis

The collected data was analyzed using both quantitative and qualitative methods. Following is the description of the techniques that was used to analyze the data in answering the research questions.

The quantitative data that was collected for this study included students’ responses to the pre- and post-test, which consisted of two multiple-choice assessments and a short attitudinal survey. Winsteps 4.0 (Linacre, 2017) software was used to conduct Rasch analysis (Rasch, 1980) and generate log odd units (logit) for students’ scores on the quantitative
measures. Rasch modeling is based on a unidimensional probabilistic model in which a subject’s probability of providing a correct response to a particular question in a test is determined by the difference between the student’s latent trait (e.g., understanding of force and motion concepts) and the difficulty of the question asked (Fox & Jones, 1998). Unlike Classical Test Theory (CTT), Rasch analysis takes into account the difficulty of the assessment items when generating student scores in terms of log odd units, which provides a more accurate measurement of the latent trait (e.g., conceptual understanding) (Boone, Townsend, & Staver, 2011). IBM SPSS Statistics 23 was used to run the descriptive statistics as well as statistical tests. After the data collection was completed, the researcher downloaded the pre/post test data from the Qualtrics website in the Microsoft Excel format, and then performed data cleaning and formatting procedures. Next, the data was transferred to the Winsteps software for Rasch analysis and students’ scores for the multiple-choice assessments and the attitudinal survey were generated on the Rasch scale. Then, IBM SPSS Statistics 23 software was used to conduct the statistical data analysis. Only the student scores on the Rasch scale was used for the statistical data analysis. Internal consistency of the multiple-choice assessments and the survey items was calculated using Cronbach’s alpha (Cronbach, 1951). Cronbach’s alpha was found to be $\alpha = 0.772$ for the CTt, $\alpha = 0.603$ for the Force and Motion Assessment, and $\alpha = 0.892$ for the attitudinal survey. Descriptive statistics for students’ raw scores as well as scores on the Rasch scale (logits) on the multiple-choice assessments and attitudinal survey were generated for both the pre-test and post-test. Also, the normality of the data was assessed both visually and statistically by using Q-Q plots and running the Shapiro-Wilk tests. The results indicated that the data were normally distributed
with significance values for the Shapiro-Wilk tests were greater than 0.05. Cohen's $d$ (Cohen, 1992) was calculated as a measure of effect size for determining the magnitude of the treatment effect that was assessed using paired-samples and independent-samples t-test analyses.

The qualitative data that was collected for this study included students’ written responses to the online reflective statement questions, classroom observation notes, and interviews with students. Students’ written responses to the reflective statement questions were downloaded from the Qualtrics website in Microsoft Excel format for qualitative data analysis and coding. The classroom observation notes and audio records of the student interviews were transcribed verbatim into Microsoft Word documents for qualitative data analysis. Coding was carried out by two researchers, the author and another doctoral student. First, both researchers carefully read the students’ responses and developed their own categories individually using an open coding approach (Creswell, 2012). Following this initial coding process, the researchers met to discuss and compare the emerging categories. During this process, the researchers looked for similarities and differences between the categories and further merged and recoded several categories, which led to the emergent themes. After the themes were determined by the coders, the author coded all responses using the themes while the second coder coded 20% of all responses. Overall, a high level of agreement (85%) was found between the two coders, $κ = .786$ (95% CI, .752 to .820, $p < .001$).
**Research Question 1**

To answer the first research question, paired-samples t-tests were used to determine if there were any statistically significant differences between students’ pre-test and post-test scores for the 16-item Force and Motion Assessment. An independent-samples t-test was conducted comparing the change in pre- and post-test scores on the Force and Motion Assessment between male and female students to investigate if the science learning gains differed significantly between the two gender groups. In addition to the pre/post test results, the following data sources were qualitatively analyzed for emergent themes about patterns of students’ conceptual understanding of force and motion concepts: (1) student responses to the reflective statement questions in Day 5 and 6 (i.e., the days students engaged in simulation-based model building activities), (2) the transcripts of the student interviews, and (3) the classroom observation notes.

**Research Question 2**

To answer the second research question, paired-samples t-tests were used to determine if there were any statistically significant differences between students’ pre-test and post-test scores for the 28-item Computational Thinking Test (CTt). An independent-samples t-test was conducted comparing the change in pre- and post-test scores on the CTt between male and female students to investigate if the computational thinking learning gains differed significantly between the two gender groups. In addition to the pre/post test results, the following data sources were qualitatively analyzed using an open coding approach for emergent themes about patterns of students’ understanding of computational thinking concepts and practices: (1) student responses to the reflective statement questions throughout
the intervention, (2) the transcripts of the student interviews, and (3) the classroom observation notes.

**Research Question 3**

Lastly, paired-samples t-tests were conducted to determine if there were any statistically significant differences between students’ pre-test and post-test scores for the attitudinal survey to answer the third research question. An independent-samples t-test was conducted comparing the change in pre- and post-test scores on the attitudinal survey between male and female students to investigate if the attitudinal gains differed significantly between the two gender groups. In addition to the pre/post test results, the following data sources were qualitatively analyzed using open coding approach for emergent themes about patterns of students’ attitudes toward programming: (1) student responses to the reflective statement questions throughout the intervention, (2) the transcripts of the student interviews, and (3) the classroom observation notes.

**Summary**

In this chapter, the convergent mixed methods design of the study was explained including the purpose and format of the intervention. Details about data sources, instruments, and data analysis were also presented.
CHAPTER FOUR: RESULTS

In this chapter, the results are presented for the three research questions. The quantitative results for each question are presented first in each section followed by the qualitative findings.

Research Question 1. How does building simulation-based computer models of physical phenomena through a visual block-based programming environment in a one-week intervention course influence middle school students’ conceptual understanding of force and motion concepts?

a. Is there a differential impact of the classroom intervention on students’ conceptual understanding of force and motion concepts according to students’ gender?

Pre-test/Post-test. Students’ pre- and post-test scores for the Force and Motion Assessment (Alonzo & Steedle, 2009) were used to investigate their conceptual understanding of force and motion concepts before and after the intervention. Internal consistency of the test was measured using Cronbach’s alpha and was found to be $\alpha = 0.603$. Table 4.1 shows descriptive statistics of the students’ raw scores for the Force and Motion Assessment along with the equivalent of logit scores generated by the Rasch analysis, which were converted to a scale of 1 to 100 for ease of interpretation. Students’ average raw score for the Force and Motion Assessment on the post-test ($M = 6.82, SD = 2.96$) increased by 2.81 points as compared to the pre-test ($M = 4.01, SD = 2.17$), with an increase of 11.99 logits on the Rasch scale. In addition, the minimum raw score was zero (i.e., no correct
answers) and the maximum was nine on the pre-test whereas the minimum increased to two
and the maximum increased to thirteen on the post-test (Table 4.1). Figure 4.1 and Figure 4.2
shows histograms of frequencies of students’ scores on Rasch scale for the Force and Motion
assessments on the pre-test and post-test, respectively.

Table 4.1

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>Median</td>
<td>Mode</td>
<td>Min.</td>
<td>Max.</td>
</tr>
<tr>
<td>Force and Motion Assess. - Pre^a</td>
<td>4.01</td>
<td>2.17</td>
<td>3.5</td>
<td>3</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Force and Motion Assess. - Post^a</td>
<td>6.82</td>
<td>2.96</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>Force and Motion Assess. - Pre (logit)^b</td>
<td>33.99</td>
<td>10.21</td>
<td>33.33</td>
<td>30.98</td>
<td>0.17</td>
<td>53.22</td>
</tr>
<tr>
<td>Force and Motion Assess. - Post (logit)^b</td>
<td>45.98</td>
<td>10.63</td>
<td>43.60</td>
<td>40.00</td>
<td>25.17</td>
<td>68.98</td>
</tr>
</tbody>
</table>

Note. N = 82.
^aRaw scores out of 16.
^bScaled to 1-100.
Figure 4.1. Histogram of the pre-test scores of the Force and Motion Assessment

Figure 4.2. Histogram of the post-test scores of the Force and Motion Assessment
A paired-samples t-test was conducted using students’ pre- and post-test scores on the Rasch scale for the Force and Motion Assessment to determine if students’ conceptual understanding of force and motion concepts significantly differed before and after the classroom intervention. The results of the paired-samples t-test, as presented in Table 4.2, showed that participating in the intervention course resulted in a statistically significant increase in seventh-grade students’ conceptual understanding of force and motion concepts with a large effect size measured through Cohen’s $d$ ($t(81) = 9.26, p < .001, d = 1.02$).

Table 4.2

<table>
<thead>
<tr>
<th></th>
<th>Pre-test</th>
<th></th>
<th>Post-test</th>
<th></th>
<th>$t$ (81)</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force and Motion Assess. (logit)$^a$</td>
<td>33.99</td>
<td>10.21</td>
<td>45.98</td>
<td>10.63</td>
<td>9.26***</td>
<td>1.02</td>
</tr>
</tbody>
</table>

*Note. N = 82.*

$^a$Scaled to 1-100.

$p < .05, **p < .01, ***p < .001$, two-tailed.

Additionally, in order to investigate how the intervention impacted male and female students’ conceptual understanding of force and motion concepts, a paired-samples t-test was conducted to compare students’ pre- and post-test scores on the Rasch scale for the Force and Motion Assessment within each group. The results of the paired-samples t-test by gender (Table 4.3) showed that participating in the classroom invention resulted in statistically significant increases in both male ($t(43) = 5.89, p < .001, d = 0.89$) and female ($t(37) = 7.35, p < .001, d = 1.02$).
students’ conceptual understanding of force and motion concepts with large effect sizes.

Table 4.3

Results of the paired-samples t-test analyses comparing students’ pre- and post-test scores on the Force and Motion Assessment by gender

<table>
<thead>
<tr>
<th></th>
<th>Pre-test</th>
<th></th>
<th></th>
<th>Post-test</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td></td>
<td>M</td>
<td>SD</td>
<td>t</td>
<td>df</td>
<td>Cohen’s d</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force and Motion Assess. (logit)\textsuperscript{a}</td>
<td>36.29</td>
<td>9.79</td>
<td></td>
<td>46.40</td>
<td>11.02</td>
<td>5.89***</td>
<td>43</td>
<td>0.89</td>
</tr>
<tr>
<td>Female</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force and Motion Assess. (logit)\textsuperscript{a}</td>
<td>31.33</td>
<td>10.17</td>
<td></td>
<td>45.49</td>
<td>10.29</td>
<td>7.35***</td>
<td>37</td>
<td>1.19</td>
</tr>
</tbody>
</table>

Note. \(n = 44\) for male, \(n = 38\) for female.
\textsuperscript{a}Scaled to 1-100.
\(p < .05\), \(**p < .01\), \(***p < .001\), two-tailed.

Lastly, an independent-samples t-test was conducted comparing the change in pre- and post-test scores for the Force and Motion Assessment on the Rasch scale between male and female students to investigate if the science learning gains differed significantly between the two gender groups. The results, as presented in Table 4.4, showed that on average female students had a higher change in their pre- and post-test scores on the Force and Motion Assessment compared to male students, however the difference between the two groups was not statistically significant (\(t (80) = 1.58, p = 0.12\)).
Table 4.4

Results of the independent-samples t-test analysis comparing the change in pre- and post-test scores for the Force and Motion Assessment between male and female students

<table>
<thead>
<tr>
<th></th>
<th>Male</th>
<th>Female</th>
<th>t</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change in Force and</td>
<td>M</td>
<td>SD</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>Motion Assess. Score</td>
<td>10.10</td>
<td>11.39</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>(logit)²</td>
<td>14.16</td>
<td>11.88</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.58</td>
<td></td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>

Note. N = 82.
²Scaled to 1-100, Change = post-test logit score – pre-test logit score
*p < .05, **p < .01, ***p < .001, two-tailed.

Student Interviews. In addition to the pre- and post-test scores, students’ responses to the interview questions provided valuable insights about their conceptual understanding of force and motion concepts. Several themes emerged from students’ responses, and the findings are presented here using sample quotes from students’ responses to illustrate their conceptual understanding of force and motion concepts.

When students were asked about what specifically they learned new or understood better about force and motion concepts during the model building activities, their responses indicated that they acquired a better conceptual understanding of the Newton's Laws of Motion and the related concepts such as acceleration. Table 4.5 shows the themes emerged from students’ responses.
Table 4.5
*Themes from student interview responses regarding their conceptual understanding of force and motion concepts*

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
</table>
| **Conceptual understanding of the Newton’s First Law of Motion** | “I actually learned that even if you don’t have any outside force, the car will still continuously go this way. I was umm before that there was unseen just kind of force acting on the entire world and everything inside, and because of that the car would just you know gradually slow but I now know that no friction no air resistance means that it will continuously go at a constant speed in the same way because nothing is slowing it down. This makes much more sense to me now.”  
- “I thought that if there is no force acting on it like no friction on the road and also no air resistance, and if you apply a constant force like 5 N and mass 500, the thing will speed up continuously, but also if you cancel that force, even without a need for outside force it will just stop. I didn’t never know that the car would keep going and going until umm forever.”  
- “Even though you don’t apply any force, it can continue to go but its speed will be the same. It will not stop because there is no friction.” |
| **Conceptual understanding of the Newton’s Second Law of Motion** | “Also, if you have more mass you have less acceleration.”  
- “It helped me get a better understanding of how the mass affects the velocity and the acceleration because then you have to apply more force depending on the mass and that will give you a bigger velocity and acceleration depending on the force and mass.”  
- “The force applied it would make a big difference even though a little bit more force applied. Like even if it goes from like 5 to 7, it would still make a big change. You know it will go even faster and faster.” |
| **Conceptual understanding of acceleration** | “With the force applied, when I put 5 or 10 or whatever, I thought it was going to go fast instantly you know, but actually it went slowly and then gradually became faster and faster although I didn’t put more force.”  
- “I also understood better how acceleration and velocity related to each other, you know if you have high acceleration it will become faster after a few seconds, if it has less acceleration it will take more to go fast.” |
When students were shown and asked to describe the simulation-based models that they built during the classroom intervention, most of the students provided details on how the models work and elaborated on different features of the models that they built. Table 4.6 shows the emergent themes from students’ responses:
Table 4.6

*Themes from student interview responses regarding their understanding and use of models*

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
</table>
| **How the models work in terms of science content**          | • “If you add force, if you change force from zero to 10, it will move, and the velocity will keep changing, keep increasing.”  
• “So, if there is a lot of force applied to car, it will go faster, if there is not it will just go slower or stop. Also, you can change the mass, and it will affect how fast the car will go umm you know if it has more mass then you need to put more force for a particular speed.”  
• “You can apply different force and you can change the mass of it, it will go with different speeds depending on how much force and mass you applied.” |
| **Conceptual understanding of force and motion concepts**    | • “… you have to apply a certain amount of Newtons so that it can move forward, and it calculates the velocity and the acceleration which is how fast it takes to go to a speed.”  
• “Well, the thing is that there is no force acting on it, so it will keep moving with the same speed. It has no acceleration because it has no force.”  
• “Yeah, if the gravity is stronger it will pull the ball more.”  
• “Even if you stop it, I mean if you stop applying force, it will keep going because there is no friction on the road and nothing tries to stop it, so it will go with the same speed.”  
• “When it is moving you can make force zero. If you do that its acceleration will be zero and its velocity will not change. It will move like this until I increase the force again.” |
| **Using or extending the models to experiment with the targeted scientific phenomena** | • “What I actually did with this model is that I did set different variables to be the different gravities of different planets of our solar system and so using a model like this to simulate different planets of gravities is really interesting and, so I think that’s really how it explains this phenomenon is that it gives us shows us how gravity interacts with the object of a certain mass to make it fall faster slower whatever it may be.”  
• “For example, you need to know how fast let’s say like your phone will fall from seven meters and it weighs ten kilograms and gravitational acceleration is 10 meters per second squared. You can use this model to simulate how fast it will fall and whether or not it will break, let’s say you know it will break if it hits with a force of 15 Newtons. If the force exceeding 15 Newtons you know the phone will shatter.” |
When students were asked about what they would change or add to improve the models that they built, their responses suggested that they had a good understanding of the nature of the models. They talked about improving their models by adding the most "relevant details" that were ignored in the first place such as friction on the road, air resistance/friction, and so on. These details are representative of target science concepts related to a more sophisticated and nuanced understanding of the scientific model. A few of the students’ responses indicated that they did not have a good understanding of the nature of models that they built in the classroom. Table 4.7 show the emergent themes from student responses.
Table 4.7
Themes from student interview responses regarding their conceptual understanding of the nature of models in science

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
</table>
| **Models are abstractions of the real-world phenomena** | • Adding friction on the road  
  o “... and figure out how to simulate friction on this model, so we can get even a more accurate model.”  
  o “Maybe we can add a friction bar for the ground, so you can make it more realistic”  
  o “I would add friction because we always have friction in real life.”  
  o “I think I would add friction to this model because there is no friction on the road…”  
  • Adding air resistance  
  o “You are gonna have to have air resistance too. Like if you are driving and you wanna do like a virtual simulation you need it to be as accurate as possible and in real world you are gonna have air resistance.”  
  o “We can also factor in the air dynamics of the car like how air resistance applies based on how fast it is going.”  
  o “I think we can even add air resistance too because air can also slow it down you know.”  
  • Applying several forces in different directions  
  o “Instead of only one force applied on it, we can have maybe two or three forces, they can be in different directions too.”  
  o “I would also add more cars and apply them different forces and see how fast they go.”  
  • Adding the gravitational force  
  o “And I think the thing that I would add would either be finding a way to simulate the force of gravity acting on this mass or the ability to figure out how much friction is applied per kilogram of mass…”  
  o “I would also add gravity. You cannot see the gravity here.” |

| **Models are replications of the real-world phenomena** | • “... also maybe if there were stop signs depending on maybe different directions that people could go…”  
  • “It is only like two things: road and a car. Maybe you can add a little more into the background. I would add more cars, houses, traffic lights maybe.”  
  • “Normal roads have speed limits, so we would make it stop when it gets like maybe a velocity of 60.”  
  • “Yeah, you are gonna have to do all the things are applied to the car in real world.” |
When asked about whether engaging in modeling activities helped them better understand Newton’s Laws of Motion, almost all students said yes and provided explanations for why they learned more. Three themes emerged from their responses: they learned more because (1) the activities were more engaging and hands-on, (2) they were able to see and visualize how the Newton’s Laws work in real time, and (3) the models were interactive, and they were able to experiment with them. Table 4.8 shows sample student responses for each theme.
Table 4.8  

Themes from student interview responses regarding their views on how models that they built helped them better understand force and motion concepts

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
</table>
| The activities were more engaging and hands-on                      | • “It did help me learn more because this was a **lot more fun than what I usually do in science**, it made me remember these things ...”  
• “This is **more engaging** than just reading it and listening to the teacher and taking notes from the slides.”  
• “If you do **hands-on** activities in science it will be more fun, and you understand more than like just writing on paper. I like to do hands-on stuff in science.” |
| They were able to see and visualize how the Newton’s Laws work in real time | • “I think it really helped me understand Newton’s Laws better because **before I knew only what the laws say**, I knew a basic grasp of what they mean and how they affected us, but I wasn’t completely sure how it actually looks like in a real-life situation. **Building this Scratch program gave an example of how an actual real-life situation look like when Newton’s laws are in action.**”  
• “Yes, I think it helped me learn it much better because I **was able to see what happens** to the car when you apply different forces also change its mass, so you know **you can understand better by looking at how it goes.**”  
• “I agree it helped a lot because you can see like Newton’s Laws in **action** like while you are playing it through changing force and mass.”  
• “**You know see like how it actually works** instead of just reading how it actually works because you don’t really understand what happens.” |
| The models were interactive, and they were able to experiment with them. | • “I have to agree it definitely made it easier to understand because I was able to actually see what was going on and **kind of experiment with this phenomenon in different ways** and ... that instead of reading it I could see it, **experiment with it, work with it until I can understand more about it than I did before.””  
• “Yeah, it does actually because I know that -umm it is actually cool to build and **play with it** because I wanna know when it goes really fast, or just keep going, or stop, and how the force plays a role in that.”  
• “I agree it helped a lot because you can see like Newton’s Laws in **action like while you are playing it through changing force and mass.”” |
In addition, two students stated that engaging in the model building activities during the classroom intervention helped them remember the mathematical equations (an alternative modeling form) regarding force and motion concepts that they studied in the previous semester: “When we were doing the program, I realized that I forgot the equations, so it helped me remember them and how to put them together.”, “This helped me remember the equations we learn last semester and putting them together you know like calculating acceleration.”

Reflective Statement Responses. At the end of each class period during the intervention, students responded online to the reflective statement question “What did I learn today?”. Students engaged in the modeling activities in Day 5 and Day 6 by building simulation-based models of the scientific phenomena. Students’ responses to the reflective statement question for Day 5 and Day 6 were analyzed qualitatively using open coding approach and the following themes emerged regarding their conceptual understanding of force and motion concepts. Table 4.9 shows the emergent themes from student responses.
Table 4.9

Themes from student reflective statement responses regarding their conceptual understanding of force and motion concepts

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
</table>
| **Conceptual understanding of the science concepts addressed in the activities** | • “I learned again about the three laws of Newton. Learning how to make a car move in scratch. The first Law of Newton was an object will be at rest or keep moving unless a force is applied to it.”  
• “I learned how acceleration changes when force changes. how to make car move at different forces. I thought car would stop when no force but it didn't because [sic] newton's law.”  
• “I learned how to make the car keep on going and bring it back to its original space. I also learned how to change the violisty [sic] and mass and more.”  
• “That I am able to add force and motion to making games. Also that even if force is not applied the car can still move.”  
• “I learned about mass and velocity. Also about gravitational acceleration and how it helps something fall.”  
• “I learned how to make a car move and, when you put the force applied on zero, it would continue to go the same velocity. we got to try to build the coding by ourselves.”  
• “I know that weight changes but mass always stays the same. We learned how to make a ball move with different types of forces.” |
| **Modeling the science concepts addressed in the activities** | • “I learned how to make a simple program that models the velocity, acceleration, and mass of a car. At first it was a little confusing but then, after some help I started to understand more. We used what we previously learned in science to help us with programming.”  
• “today i learned how to use speed, velocity, acceleration, and mass for a car on the website scratch. I also learned how we can increase the velocity and how it accelerates and also controlling its speed.”  
• “I learned how to apply force and motion to my program to help me understand newtons laws of motion.”  
• “I learned today how to create a scientific computer model that can simulate the forces of gravity on an object.”  
• “I learned how to make a program that took into account velocity, weight, mass, and gravitational acceleration. I used to not be able to explain the code, but at the end of the class I was able to go up and explain it to everyone. I really enjoyed learning more about coding.”  
• “I learned how to make a ball fall down in a world that has no air resistance in Scratch. I can now make scientific models in scratch.” |
**Research Question 2.** How does participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment influence middle school students’ understanding of computational thinking concepts and practices?

a. Is there a differential impact of the classroom intervention on students’ understanding of computational thinking concepts and practices according to students’ gender?

**Pre-test/Post-test.** Students’ pre- and post-test scores for the CTt were used to investigate their understanding of computational thinking concepts and practices before and after the intervention. Internal consistency of the test was measured using Cronbach’s alpha and was found to be $\alpha = 0.772$. Table 4.10 shows descriptive statistics of students’ raw scores for the CTt along with the equivalent of logit scores generated by Rasch analysis, which were converted to a scale of 1 to 100 for easy interpretation. Students’ average raw score for the CTt on the post-test ($M = 17.83$, $SD = 5.63$) increased by 2.93 points as compared to the pre-test ($M = 20.76$, $SD = 4.29$), with an increase of 7.92 logits on the Rasch scale. Figure 4.3 and Figure 4.4 shows histograms of frequencies of students’ scores on Rasch scale for the CTt on the pre-test and post-test, respectively.
Table 4.10

*Descriptive statistics for students’ scores on the CTt*

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>SD</th>
<th>Median</th>
<th>Mode</th>
<th>Min.</th>
<th>Max.</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTt-Pre(^a)</td>
<td>17.83</td>
<td>5.63</td>
<td>19</td>
<td>21</td>
<td>6</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td>CTt-Post(^a)</td>
<td>20.76</td>
<td>4.29</td>
<td>22</td>
<td>22</td>
<td>11</td>
<td>28</td>
<td>17</td>
</tr>
<tr>
<td>CTt-Pre (logit)(^b)</td>
<td>58.37</td>
<td>12.77</td>
<td>58.98</td>
<td>63.11</td>
<td>34.12</td>
<td>99.93</td>
<td>65.81</td>
</tr>
<tr>
<td>CTt-Post (logit)(^b)</td>
<td>65.33</td>
<td>11.05</td>
<td>66.22</td>
<td>66.22</td>
<td>45.06</td>
<td>99.97</td>
<td>54.91</td>
</tr>
</tbody>
</table>

*Note. N = 82.*

\(^a\) Raw scores out of 28.

\(^b\) Scaled to 1-100.

---

*Figure 4.3. Histogram of the pre-test scores of the CTt*
A paired-samples t-test was conducted using students’ pre- and post-test logit scores for the Computational Thinking Assessment to determine if students’ understanding of computational thinking concepts and practices significantly differed before and after the classroom intervention. The results of the paired-samples t-test, as presented in Table 4.11, showed that participating in the intervention course resulted in a statistically significant increase in seventh-grade students’ understanding of computational thinking concepts and practices with a large effect size measured through Cohen's $d$ ($t(81) = 8.06, p < .001, d = 0.91$).
Additionally, in order to investigate how the intervention impacted male and female students’ understanding of computational thinking concepts and practices, a paired-samples t-test was conducted to compare students’ pre- and post-test scores for the CTt on the Rasch scale within each group. The results of the paired-samples t-test by gender, as presented in Table 4.12, showed that participating in the classroom invention resulted in statistically significant increases in both male ($t(43) = 3.68, p < .01, d = 0.56$) and female ($t(37) = 10.99, p < .001, d = 1.80$) students’ understanding of computational thinking concepts and practices.

### Table 4.11

**Results of the paired-samples t-test comparing students’ pre- and post-test scores on the CTt**

<table>
<thead>
<tr>
<th></th>
<th>Pre-test</th>
<th></th>
<th>Post-test</th>
<th></th>
<th></th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M$</td>
<td>$SD$</td>
<td>$M$</td>
<td>$SD$</td>
<td>$t$</td>
<td></td>
</tr>
<tr>
<td>CTt (logit)$^a$</td>
<td>58.37</td>
<td>12.77</td>
<td>65.33</td>
<td>11.05</td>
<td>8.06***</td>
<td>0.91</td>
</tr>
</tbody>
</table>

*p < .05, **p < .01, ***p < .001, two-tailed.*

Note. $N = 82$.

$^a$Scaled to 1-100.

### Table 4.12

**Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the CTt by gender**

<table>
<thead>
<tr>
<th></th>
<th>Pre-test</th>
<th></th>
<th>Post-test</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M$</td>
<td>$SD$</td>
<td>$M$</td>
<td>$SD$</td>
<td>$t$</td>
<td>df</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTt (logit)$^a$</td>
<td>61.81</td>
<td>13.40</td>
<td>66.95</td>
<td>11.72</td>
<td>3.68**</td>
<td>43</td>
</tr>
<tr>
<td>Female</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTt (logit)$^a$</td>
<td>54.38</td>
<td>10.84</td>
<td>63.45</td>
<td>10.06</td>
<td>10.99***</td>
<td>37</td>
</tr>
</tbody>
</table>

Note. $n = 44$ for male, $n = 38$ for female.

$^a$Scaled to 1-100.

*p < .05, **p < .01, ***p < .001, two-tailed.*
Lastly, an independent-samples t-test was conducted comparing the change in pre- and post-test scores for the CTt on the Rasch scale between male and female students to investigate if the computational thinking learning gains differed significantly between the two gender groups. The results, as presented in Table 4.13, showed that on average female students had a higher change in their pre- and post-test scores on the CTt compared to male students and the difference between the two groups was statistically significant ($t(80) = 2.34$, $p < .05$).

<table>
<thead>
<tr>
<th></th>
<th>Male</th>
<th></th>
<th>Female</th>
<th></th>
<th>t</th>
<th>df</th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change in CTt Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(logit)$^a$</td>
<td>$M$ 9.13</td>
<td>$SD$ 9.25</td>
<td>$n$ 44</td>
<td>$M$ 9.07</td>
<td>$SD$ 5.09</td>
<td>$n$ 38</td>
<td>$t$ 2.34 *</td>
</tr>
</tbody>
</table>

*Note. N = 82.*

$^a$Scaled to 1-100, Change = post-test logit score – pre-test logit score

*p < .05, **p < .01, ***p < .001, two-tailed.

**Student Interviews.** In addition to the pre- and post-test scores, students’ responses to the interview questions provided in-depth insights about their understanding of computational thinking concepts and practices introduced in the classroom intervention. A number of sample quotes from students’ responses are presented here to illustrate their understanding of computational thinking concepts and practices.
One student explained how he was able to give instructions to the objects in his model for making them to do certain actions and calculate the variables by using conditional and loops structures:

Also, the main aspect of the class I liked was using the equations we learned in science like how to calculate force, mass, velocity and equations like that in applying into a game to give an object or a character or the car for instance in this model decisions on what to do like using if-else and forever loop.

When asked about what was the most challenging part of the model building activities, another student explained the importance of making sure of choosing the correct loop structure when programming the model, which provides evidence for his understanding of different loop structures (i.e., forever, repeat a certain number of times, repeat until a condition is satisfied) in programming: “The most challenging part was to get the car to go to a certain umm like when you made the force a lot sometimes it will not move because you made a little mistake like using forever loop instead of other loop”.

A female student explained why she enjoyed learning about programming by talking about how she used variables and loop structures to create models, which provides evidence of her understanding of abstractions by creating and using variables to represent physical quantities and imbedding and using these variables in programming and building models of physical phenomena: “I did enjoy learning about programming a lot and the aspects that I really enjoyed the most was learning about how to use variables in conjunction with loops to create really interesting programs and models, and I think that that’s really what stuck to me”.

84
**Reflective Statement Responses.** Students engaged in programming activities throughout the intervention, either in learning programming or in building simulation-based models, and their responses to the reflective statement questions were analyzed qualitatively using the previously described open coding approach. Table 4.14 presents the themes emerged regarding their understanding of computational thinking concepts and practices.
Table 4.1
Themes from student reflective statement responses regarding their understanding of computational thinking concepts and practices

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
</table>
| Code as a set of instructions for the agents to execute | • “I learned how to put the programming blocks together to create a program or even just an action that the avatar could execute.”  
• “Today I learned how to code the cat to make it move and make the tennis ball move and respond to directions.”  
• “we learn how to control the cat on different ways.”  
• “I learned how to make something move in a certain direction at a certain time.”                                                                                                                                                                                               |
| Conditionals – using decision making in programming     | • “Today I learned to program a Pong game and how to make decisions. We made or cat make a decision by the prices, which was very cool!”  
• “I learned more about loops and of the if this then that. It was fun and it was also about the green blocks.”  
• “I have learned what a if or else command and i have learned how to make a pong game i have learned how to be a better programmer”  
• “More programming and more choice making.”  
• “that to take out many options or commands, then you would use the if then, else block”                                                                                                                                                                           |
| Loops – using repetition structures in programming      | • “I learned how to use loops and conditions to run separate pieces of code over and over”  
• “We learned how to make loops using repeat # times, repeat until, and forever.”  
• “I learned how to make something do something for some amount of time and until certain conditions are met.”  
• “I learned about loops like repeating blocks that repeat the script inside that block how many ever times it wants.”                                                                                                                                                     |
| Variables – creating and using variables in programming | • “I learned how to create and use variables in Scratch. I can also use the broadcast command to send messages.”  
• “We learned about variables. It is made by a person and can be activated my different things. Also we learned how to broadcast.”  
• “I learned how to deal with variables and broadcast messages to the other sprites. I didn't learn about this in computer science, so it was interesting to learn about it today.”  
• “Today in class I learned more about how to scratch. The main thing he focused on today was variables. He taught us how to make a die roll thing use a random variable to determine the outcome.” |
<table>
<thead>
<tr>
<th>Table 4.1 (continued).</th>
<th></th>
</tr>
</thead>
</table>
| Nesting – embedding programming instructions (blocks) inside one another | • “I know all of this but I mean I learned that you can put a loop in a loop.”  
• “I learnt [sic] loops and how to put one inside the other to do some cool things.”  
• “Loops that reepeat [sic] the things, and also nested loops and conditionals. we built program to look at two things at the same time. cool stuff”  
• “today i learned how to use a forever loop and a long loop. i also learned how to do a nested thing.”  
• “I learned how to put if into another if. I leanred [sic] how the cat falls in scratch. loops also” |
| Event-driven programming – using broadcast messages to trigger actions in different scripts | • “I learned i can make a broadcast to other characters and it will move when it receives the broadcast. you can make a dice roller.”  
• “that we can send messages to other sprites and they can or do the things we ask them for”  
• “I learned how to broadcast and create animations with 2 characters.”  
• “Broadcasting sends messages to others that will receive it. It will then do an action until fits done.”  
• “how to broadcast a signal to other sprites on the stage. The signal broadcast tells the sprite what to do.”  
• “I learned how to broadcast directions and signals to other characters, this made them do certain actions. I also learned how to make a game.”  
• “I learned how to broadcast a command and make something do something once it received a command.”  
• “I learned how to make the sprites signal to each other to do things. I also learned how to make dice roll and chose random numbers.” |
| Scratch environment – understanding Scratch’s programming syntax and user interface | • “I learned all types of different coding blocks. Also learned how to put certain blocks together, and how they work. The basics of coding on scratch.”  
• “I learned how to make a remix about somebody else's project.”  
• “How to work with the programming website scratch and work with different situations.”  
• “I learned how to use scratch more. Also we made a ping pong game with the code that scratch has. Also how the code that scratch has is like regular code.”  
• “Today I learned the basics of how to code using scratch command blocks. I also learned how to make a simple game of ping pong. The lesson was really fun and easy to learn.” |
Research Question 3. How do middle school students’ attitudes towards programming change as a result of participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment?

a. Is there a differential impact of the classroom intervention on students’ attitudes towards programming according to students’ gender?

Pre-test/Post-test. Students’ pre- and post-test scores for the attitudinal survey were used to investigate their attitudes towards programming before and after the classroom intervention. Internal consistency of the survey was measured using Cronbach’s alpha and was found to be \( \alpha = 0.892 \). Table 4.13 shows descriptive statistics of the students’ aggregated raw scores for the attitudinal survey along with the equivalent of logit scores generated by Rasch analysis, which were converted to a scale of 1 to 100 for easy interpretation. Students’ average aggregated raw score for the attitudinal survey on the post-test \((M = 3.18, SD = 0.78)\) increased by 0.33 points as compared to the pre-test \((M = 3.51, SD = 0.73)\), with an increase of 8.50 logits on the Rasch scale. Figure 4.5 and Figure 4.6 shows histograms of frequencies of students’ scores on Rasch scale for the attitudinal survey on the pre-test and post-test, respectively. In addition to the aggregated scores presented in Table 4.15, Table 4.16 shows the mean, standard deviation, minimum and maximum of students’ raw scores for each item on the attitudinal survey for the pre- and post-test. An examination of Table 4.16 reveals that students’ mean raw scores for each item on the attitudinal survey increased on the post-test compared to the pre-test, with the minimum scores for two items (item 1 = “I’m sure of
myself when I do programming”, item 6 = “I know I can do well in programming”) increased from 1 (i.e., “Strongly Disagree”) to 2 (i.e., “Disagree”) on the post-test.

Table 4.15
*Descriptive statistics for students’ scores on the attitudinal survey*

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>SD</th>
<th>Median</th>
<th>Mode</th>
<th>Min.</th>
<th>Max.</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attitudinal Survey-Pre</td>
<td>3.18</td>
<td>0.78</td>
<td>3.25</td>
<td>3.25</td>
<td>1.38</td>
<td>5.00</td>
<td>3.63</td>
</tr>
<tr>
<td>Attitudinal Survey-Post</td>
<td>3.51</td>
<td>0.73</td>
<td>3.5</td>
<td>3.38</td>
<td>2.00</td>
<td>5.00</td>
<td>3.00</td>
</tr>
<tr>
<td>Attitudinal Survey-Pre (logit)</td>
<td>49.37</td>
<td>13.29</td>
<td>49.27</td>
<td>49.27</td>
<td>21.24</td>
<td>99.95</td>
<td>78.71</td>
</tr>
<tr>
<td>Attitudinal Survey-Post (logit)</td>
<td>57.87</td>
<td>14.20</td>
<td>56.31</td>
<td>54.18</td>
<td>32.28</td>
<td>99.96</td>
<td>67.68</td>
</tr>
</tbody>
</table>

*Note. N = 82.*

*a*Eight items on a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree)

*b*Scaled to 1-100.

*Figure 4.5. Histogram of the pre-test scores of the Attitudinal Survey*
Figure 4.6. Histogram of the post-test scores of the Attitudinal Survey
Table 4.16

Descriptive statistics for students’ average scores on each survey item

<table>
<thead>
<tr>
<th>Survey Item</th>
<th>Pre-test</th>
<th>Post-test</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I am sure of myself when I do programming.</td>
<td>3.35 (0.98)</td>
<td>3.63 (0.91)</td>
<td>1 (5)</td>
<td>2 (5)</td>
</tr>
<tr>
<td>2. I would consider a career in programming.</td>
<td>2.67 (1.14)</td>
<td>3.28 (1.02)</td>
<td>1 (5)</td>
<td>1 (5)</td>
</tr>
<tr>
<td>3. I expect to use programming when I get out of school.</td>
<td>3.11 (1.12)</td>
<td>3.49 (0.97)</td>
<td>1 (5)</td>
<td>1 (5)</td>
</tr>
<tr>
<td>4. Knowing programming will help me earn a living.</td>
<td>3.48 (1.02)</td>
<td>3.73 (0.91)</td>
<td>1 (5)</td>
<td>1 (5)</td>
</tr>
<tr>
<td>5. I will need programming for my future work.</td>
<td>3.11 (1.08)</td>
<td>3.27 (0.98)</td>
<td>1 (5)</td>
<td>1 (5)</td>
</tr>
<tr>
<td>6. I know I can do well in programming.</td>
<td>3.44 (1.10)</td>
<td>3.88 (0.80)</td>
<td>1 (5)</td>
<td>2 (5)</td>
</tr>
<tr>
<td>7. Programming will be important to me in my life’s work.</td>
<td>3.04 (1.04)</td>
<td>3.21 (0.93)</td>
<td>1 (5)</td>
<td>1 (5)</td>
</tr>
<tr>
<td>8. I am sure I could do advanced work in programming.</td>
<td>3.24 (1.18)</td>
<td>3.59 (1.13)</td>
<td>1 (5)</td>
<td>1 (5)</td>
</tr>
</tbody>
</table>

Note. N = 82. All items are on a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree).

A paired-samples t-test was conducted using students’ pre- and post-test aggregated logit scores for the attitudinal survey to investigate whether students’ attitudes towards programming significantly differed before and after the classroom intervention. The results of the paired-samples t-test, as presented in Table 4.17, showed that participating in the intervention course resulted in a statistically significant increase in seventh-grade students’ attitudes towards programming with a moderate effect size (t (81) = 5.93, p < .001, d = 0.66).
Table 4.17

Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the attitudinal survey

<table>
<thead>
<tr>
<th>Attitudinal Survey (logit)a</th>
<th>Pre-test</th>
<th>Post-test</th>
<th>t (81)</th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td></td>
<td>49.37</td>
<td>13.29</td>
<td>57.87</td>
<td>14.20</td>
</tr>
</tbody>
</table>

Note. N = 82.
aScaled to 1-100.
*p < .05, **p < .01, ***p < .001, two-tailed.

Additionally, in order to investigate how the intervention impacted male and female students’ attitudes towards programming, a paired-samples t-test was conducted to compare students’ pre- and post-test aggregated logit scores for the attitudinal survey within each group. The results of the paired-samples t-test by gender, as presented in Table 4.18, showed that participating in the classroom invention resulted in statistically significant increases in both male (t (43) = 4.12, p < .001, d = 0.62) and female (t (37) = 4.24, p < .001, d = 0.69) students’ attitudes towards programming with moderate effect sizes.

Table 4.18

Results of the paired-samples t-test analysis comparing students’ pre- and post-test scores on the attitudinal survey by gender

<table>
<thead>
<tr>
<th></th>
<th>Pre-test</th>
<th>Post-test</th>
<th>t</th>
<th>df</th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male Attitudinal Survey (logita)</td>
<td>55.29</td>
<td>13.14</td>
<td>62.84</td>
<td>13.76</td>
<td>4.12***</td>
</tr>
<tr>
<td>Female Attitudinal Survey (logita)</td>
<td>42.53</td>
<td>9.84</td>
<td>52.11</td>
<td>12.57</td>
<td>4.24***</td>
</tr>
</tbody>
</table>

Note. n = 44 for male, n = 38 for female.
aScaled to 1-100.
*p < .05, **p < .01, ***p < .001, two-tailed.
Lastly, an independent-samples t-test was conducted comparing the change in pre- and post-test scores for the attitudinal survey between male and female students to investigate if the change in attitudes differed significantly between the two gender groups. The results, as presented in Table 4.19, showed that on average female students had a higher change in their pre- and post-test scores on the attitudinal survey compared to male students, however the difference between the two groups was not statistically significant ($t(80) = 0.702, p = 0.485$).

Table 4.19  
*Results of the independent-samples t-test analysis comparing the change in pre- and post-test scores for the attitudinal survey between male and female students*

<table>
<thead>
<tr>
<th></th>
<th>Male</th>
<th></th>
<th></th>
<th>Female</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>n</td>
<td>M</td>
<td>SD</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>
| Change in Attitudinal    | 7.55 | 12.17         | 44         | 9.58   | 13.92         | 38         | 0.702| 80   
| Survey Score (logit)a    |      |               |            |        |               |            |      |  

*Note. N = 82.

*aScaled to 1-100, Change = post-test logit score – pre-test logit score

*p < .05, **p < .01, ***p < .001, two-tailed.*

**Student Interviews.** In addition to the results of the pre- and post-test attitudinal survey, students’ responses to the interview questions were qualitatively analyzed using the open coding approach described above to provide an in-depth analysis of their attitudes towards programming as well as the overall intervention in the classroom. All the students participated in the interviews at the end of the intervention appeared to have very positive attitudes towards learning about programming. Several quotes of student responses were provided in Table 4.20 regarding their positive attitudes towards programming.
Table 4.20

Quotes from student interview responses regarding their positive attitudes towards programming

- “I enjoyed learning computer programming because now I can make the characters actually do things what I want them to and make things move. I can now actually make those things and change in any way I want because I know programming. And what I liked most was that I can actually make the characters talk you know with bubbles. It was pretty fun.”
- “I also enjoyed programming. It was fun. I think the activities were good. It was all hands-on and you know we were actually doing programming like building a game, pong game. I think it was very engaging. I can now make science animations, like make car move with different forces.”
- “I really enjoyed learning programming, it was fun like to put the blocks and see how it works. You can play around with it and see what works better and then if you have any mistakes you can do better in future.”
- “I enjoyed a lot. I didn’t know programming before, so, it was fun to learn how you can put different blocks on things to see how you can make other things maybe move or talk or dance.”
- “I enjoyed trying to figure out on our own what we were supposed to do for the code and like how it was gonna work because sometimes it didn’t work so it was kind of trial and error for me. And I work better when I’m able to do it on my own without having to copy something.”
- “I just enjoyed it a lot. It was really fun to learn programming and learning it with science. It was more challenging this way, but it was better than what we normally do in Ms. Smith’s [pseudonym] class.”
- “I liked the programming. My favorite thing to do in class is to do hands-on activities. It is what I like to do in classes.”
- “I enjoyed it a lot because all the work you have done you can see and play out and see how it works and modify it and do stuff. It was really cool. I liked seeing all the work I’ve done. I can see how it is actually working because I did this. It’s pretty cool.”

Reflective Statement Responses. Students’ responses to the reflective statement questions throughout the classroom intervention were also analyzed qualitatively using open
coding approach to understand their attitudes towards programming as well as the classroom intervention, and the emerging themes were presented in Table 4.21.

A lot of students showed highly positive attitudes towards programming and the classroom intervention in their responses to the reflective statement questions. However, a very small number of students appeared to have negative attitudes (Table 4.21). Two of these students found block-based programming very difficult. Another student who knew programming before attending the intervention and wanted to learn a more advanced programming language such as Python found the learning activities to be boring.
Table 4.21
Themes from student reflective statement responses regarding their attitudes towards programming

<table>
<thead>
<tr>
<th>Themes</th>
<th>Sample Student Quotes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positive Attitude</strong></td>
<td>• “I learned about block coding and how to make your program a cool one. <em>It was a cool experience.</em>”</td>
</tr>
<tr>
<td></td>
<td>• “Today I learned to program a Pong game and how to make decisions. We made or cat make a decision by the prices, <em>which was very cool!</em>”</td>
</tr>
<tr>
<td></td>
<td>• “that you have to doa [sic] lot of work and that <em>its fun just playing around with it</em>”</td>
</tr>
<tr>
<td></td>
<td>• “I learned how programming works and did it <em>which was fun.</em>”</td>
</tr>
<tr>
<td></td>
<td>• “I learned how to program more effectively an [sic] had a lot of fun doing programs.”</td>
</tr>
<tr>
<td></td>
<td>• “I learned how to do basic coding and and I <em>had a fun time</em>”</td>
</tr>
<tr>
<td></td>
<td>• “Today I learned the basics of how to code using scratch command blocks. I also learned how to make a simple game of ping pong. <strong>The lesson was really fun and easy to learn.</strong>”</td>
</tr>
<tr>
<td></td>
<td>• “i learned that you can make any game by just putting alittle [sic] of steps and <em>its really cool</em>”</td>
</tr>
<tr>
<td><strong>Negative Attitude</strong></td>
<td>• “i learned that coding is <em>very hard</em> so i don't really like it. But i guess if it was something i had to do”</td>
</tr>
<tr>
<td></td>
<td>• “Again nothing. I'm glad this the last day because this class was <em>so boring.</em> These blocks aren't programming. Teach me python or Java script.”</td>
</tr>
<tr>
<td></td>
<td>• <strong>Nothing that will benefit me in life</strong>”</td>
</tr>
<tr>
<td></td>
<td>• “nothing because it was <em>very difficult</em> and I didn’t really like”</td>
</tr>
</tbody>
</table>

**Summary**

This chapter presented the results of the quantitative and qualitative data analysis that are organized around the research questions. The next chapter will provide a discussion of these results and give both theoretical and practical explanations regarding the findings of the study.
CHAPTER FIVE: DISCUSSION AND IMPLICATIONS

The results from the quantitative and qualitative measures for each research question were reported in the previous chapter. This chapter attempts to discuss the results pertaining to each research question and provide explanations of the findings. In addition, implications of the results for classroom practice as well as for future research are discussed in this chapter.

Research Question 1. How does building simulation-based computer models of physical phenomena through a visual block-based programming environment in a one-week intervention course influence middle school students’ conceptual understanding of force and motion concepts?

a. Is there a differential impact of the classroom intervention on students’ conceptual understanding of force and motion concepts according to students’ gender?

During the classroom intervention, students engaged in learning about and constructing computational abstractions in the form of simulation-based models to demonstrate and further their conceptual understanding of force and motion concepts. The Force and Motion Assessment developed by Alonzo and Steedle (2009) was used as a pre- and post-test to assess students’ conceptual understanding of force and motion concepts prior to and after the classroom intervention. In addition, students’ responses to the reflective statement questions at the end of each class period as well as their responses to the interview
questions provided in-depth, triangulating insights regarding what students learned about force and motion concepts during the intervention.

Participating students had formally studied the force and motion unit in the classroom in the previous semester (i.e., Fall 2016). However, the results of the pre-test (Table 4.1) suggest that students’ current conceptual understanding of force and motion concepts was very limited before they participated in the classroom intervention of this study ($M = 33.39$, $SD = 10.21$; out of 100 on Rasch scale). This finding aligns with the findings of previous studies showing that students held various alternative conceptions about force and motion concepts that were highly persistent and resistant to change, and traditional, lecture-based science instruction generally failed to address those alternative conceptions in the classroom (cf., Eryilmaz, 2002; Reiner, Slotta, Chi, & Resnick, 2000; Dykstra & Sweet, 2009; Hestenes, 1992a; Ioannides & Vosniadou, 2001; Neumann et al., 2013).

This study aimed to foster students’ conceptual understanding of force and motion concepts by introducing computational thinking practices and engaging them in building simulation-based models of physical phenomena through a visual programming environment. The results of the paired-samples t-test comparing students’ pre- and post-test scores on the Force and Motion Assessment (Table 4.2) showed that participating in a week-long classroom intervention resulted in a statistically significant increase in students’ conceptual understanding of force and motion concepts with a large effect size. Students’ mean score for the Force and Motion Assessment on the post-test ($M = 45.98$, $SD = 10.63$; out of 100 on Rasch scale) increased by 12.59 logits compared to their mean score on the pre-test ($M = 33.39$, $SD = 10.21$; out of 100 on Rasch scale). Despite significant learning gains during the
intervention, the results also indicate that students on average did not demonstrate a high level of understanding of force and motion concepts on the post-test, suggesting that even more time may be needed in the classroom to address some common alternative conceptions students might still hold through additional modeling activities. It is important to restate here that students learned about computational thinking concepts and practices in the first three days (i.e., three class periods) of the intervention and then engaged in the model building activities around force and motion concepts for only two days (i.e., two class periods). So, the results are promising, in that this pedagogical approach of infusing computational thinking concepts and practices into science instruction in the context of building computational abstractions (i.e., simulation-based models) of physical phenomena can promote conceptual science learning in the classroom with only a minimal amount of classroom time. These findings agree with the findings of a limited, yet growing, number studies in the literature showing that incorporating computational thinking concepts and practices into science instruction in the context of generating computational models can promote conceptual science learning (Basu et al., 2012; diSessa, 2000; Sengupta & Farris, 2012; Wilensky, Brady, & Horn, 2014; Wilkerson-Jerde et al., 2015). For example, in a study with sixth-grade students, Basu and colleagues (2012) developed a computationally-enabled learning environment called CTSiM (“Computational Thinking in Simulation and Modeling”) that let students construct computational models using a built-in visual programming interface. Students participated in a six-day classroom intervention where they engaged in modeling real-world phenomena using CTSiM to learn about kinematics and ecology. Similar to the findings of this study, results of their pre- and post-test assessment
showed significant learning gains in both science topics (Basu et al., 2012). It is important to note here that both Basu and colleagues (2012) and the present study employed immediate post-test to assess student learning. However, this approach does not confirm the lasting effects of the classroom intervention of this study on students’ conceptual learning of force and motion concepts.

Previous research showed that gender may play a significant role in shaping students’ confidence and self-efficacy towards learning about computational thinking practices in general, and computer programming in particular (Duncan & Bell, 2015; Ericson & McKlin, 2012; Margolis, Fisher, & Miller, 2000; Orton et al., 2016). For instance, in a recent study with a large sample of high school students in the United States ($N = 475$), Orton and colleagues (2016) found that female students were less comfortable with using computers and computational tools compared to male students. Since the intervention of the present study included learning about computational thinking as well as working with computational tools, the researcher investigated whether there was a differential impact of the classroom intervention on students’ conceptual understanding of science concepts based on gender by running a paired-samples t-test to compare students’ pre- and post-test scores for the Force and Motion Assessment within each gender group. The results of the paired-samples t-test (Table 4.3) showed that both male and female students’ average scores on the post-test increased significantly compared to the pre-test with large effect sizes. The researcher did run an independent-samples t-test to investigate if there was also a significant difference between male and female students’ science learning gains. The results showed that female students on average demonstrated higher science learning gains compared to male students, however the
difference in the learning gains between male and female students was not statistically significant (Table 4.4). Overall, the results suggest that there was no differential impact of the classroom intervention on students’ conceptual understanding of force and motion concepts based on their gender. It is important to note here that female students’ average pre- and post-test scores on the Force and Motion Assessment were lower compared to male students as shown in Table 4.3. In addition, the classroom observation notes showed no discrepancy between male and female students in terms of their attitudes towards and engagement in the programming and modeling activities during the intervention. While several studies in the literature (Basu et al., 2012; diSessa, 2000; Sengupta & Farris, 2012; Wilensky et al., 2014; Wilkerson-Jerde et al., 2015) investigated how computationally-enabled model building activities can promote science learning in the classroom, no study, to date, examined whether this kind of modeling activities would have any differential impact on students’ science learning based on gender. This study contributes to the literature by suggesting that integrating computational thinking concepts and practices into science instruction through computational model building activities in middle school science classrooms does not have a differential impact on students’ science learning based on gender.

In addition to the pre- and post-test results, students’ responses to the reflective statement questions as well as their responses to the interview questions provided valuable insights about their conceptual understanding of force and motion concepts. Here is an example of a student’s response to the reflective statement question of “What did I learn today?” at the end of the first modeling activity on Day 5: “I learned how acceleration changes when force changes. how to make car move at different forces. I thought car would
stop when no force but it didn't becase [sic] newton's law”. Firstly, this student’s response indicates that she understood that acceleration of an object is dependent on how much net force is applied on the object (i.e., Newton’s Second Law of Motion). In other words, the student was able to grasp an important force and motion concepts that acceleration, not velocity, is a function of applied force, which means a change in net force results in a change in acceleration of an object. Previous research showed that students tend to reason that velocity is directly proportional to force, increasing velocity meaning increasingly applied force (Dykstra & Sweet, 2009; Hestenes, Wells, & Swackhamer, 1992). Secondly, this student’s response also suggests that she had held this conception of “motion requires force” (i.e., if there is a motion, then there must a force acting): “… I thought car would stop when no force ...”, which is a very common alternative conception among students from elementary grades to college. Students with this alternative conception often have difficulty conceptualizing a frictionless medium (diSessa & Sherin, 1998; Eryilmaz, 2002). In the first modeling activity, students engaged in building a model in Scratch that simulated the motion of a car on a frictionless road. In the second modeling activity, students engaged in building a model to simulate the fall of a basketball in a medium without air resistance. By engaging in these modeling activities, the student seemed to have revised their alternative conception and gained a better conceptual understanding of the Newton’s First and Second Law of Motion.

When asked in the interviews about what specifically they learned or understood better about force and motion concepts during the computational modeling activities, another student’s response indicated how their understanding of the Newton’s First Law changed:
I actually learned that even if you don’t have any outside force, the car will still continuously go this way, I was umm before that there was unseen just kind of force acting on the entire world and everything inside, and because of that the car would just you know gradually slow but I now know that no friction no air resistance means that it will continuously go at a constant speed in the same way because nothing is slowing it down. This makes much more sense to me now.

The student explicitly elaborated on their thinking about how the car would go in the absence of friction before and after he engaged in the modeling activities. Like most of the other students in the classroom, he thought that the car needs force to continue its motion even in the absence of friction (i.e., “motion requires force”): “… there was unseen just kind of force acting on the entire world and everything inside and because of that the car would just you know gradually slow ...”. After constructing and experimenting with his model, the student was able to conceptually grasp the idea that no force is needed for the car to go at constant velocity because “... nothing is slowing it down”.

Acceleration is a fundamental concept in physics and its conceptual understanding is essential when learning about force and motion concepts. However, numerous studies in the literature have consistently shown that acceleration is a particularly challenging concept for students to conceptually grasp (Jones, 1983; Rosenquist & McDermott, 1987; Dykstra & Sweet, 2009; Hestenes, 2007). Acceleration can be simply defined as "the time rate of change of velocity" (Serway & Jewett, 2014; p. 32). In an earlier study with middle and high school students, Jones (1983) found that the following alternative conceptions were prevalent among students: ‘as objects speed up their acceleration also increase’ and ‘objects have a nonzero acceleration even if they move with a constant high speed’. Students often confuse
acceleration with velocity or speed, thinking that if two objects have the same speed, then they should have the same acceleration (Rosenquist & McDermott, 1987; Dykstra & Sweet, 2009; Tasar, 2010). Perhaps not surprisingly, similar to the findings of the previous studies, participating students of this study struggled explaining and conceptualizing acceleration during the classroom intervention. Since force and motion unit was covered in the previous semester, some students were able to recite a definition that they had probably memorized for the exam. However, when the researcher asked probing questions to assess their conceptual understanding, most of the students were not able to explain it in their own words. In both modeling activities, students were supposed to create separate variables for acceleration and velocity of the object on the screen (i.e., car and basketball), and formulate an algorithm using computational concepts and mathematical operators available in the Scratch environment to define the relationship between the variables and then compute them, so the running model could show the values of the variables on the screen in real time. In order to formulate an algorithm, students needed to conceptually understand what is meant by acceleration and how the velocity of the object is related to and calculated from its acceleration. For most of the students, it was their “ah-ha” moment in the classroom when they were able to figure out the relationship between acceleration and velocity. The following quote from the interviews demonstrates one of the participating student’s conceptual understanding of acceleration in their own words:

With the force applied, when I put 5 or 10 or whatever, I thought it was going to go fast instantly you know because it has acceleration, but actually it went slowly and then gradually became faster and faster although I didn’t put more force. It was kind of eye-opening.
The student was talking about the first modeling activity that simulated a car’s motion with an applied force on a frictionless road. It is evident that this student conflated acceleration with velocity as they thought with a given acceleration the car will immediately go fast. However, after engaging in the modeling activity, he was able to understand that acceleration made the car “gradually became faster and faster”. Here is another student from the interviews that defined acceleration in her own words, which illustrates her deeper understanding of the concept: “I also understood better how acceleration and velocity related to each other, you know if you have high acceleration it will become faster after a few seconds, if it has less acceleration it will take more to go fast”. The student certainly did a great job describing how the magnitude of acceleration of an object affect the “rate” at which its velocity changes.

Looking at the significant learning gains on the pre- and post-test, as well as the evidence of students’ conceptual understanding of force and motion concepts from their responses to the reflective statement and interview questions, it is important to discuss how the modeling activities of the intervention facilitated conceptual science learning in the classroom. Were the significant learning gains on force and motion concepts because of the fact that participating students spent just more time in the classroom on this topic? The answer is probably “no” because the participating students had already spent several weeks on the force and motion unit in the classroom previous semester and their mean score for the Force and Motion assessment on the pre-test was \( M = 33.99 \) out of 100 on Rasch scale, which was relatively low. It is likely that what led to the learning gains among the participating students was their high cognitive engagement in the computational modeling
activities. Then, how did the modeling activities help students gain a better conceptual understanding of force and motion concepts? Drawing on the theoretical framework—Representational Construction Affordances (Prain & Tytler, 2012)—of this study, the affordances of simulation-based computational models as a dynamic form of representation were evident in students’ interview responses as they were able to “see and visualize” how the target physical phenomenon works or behaves in a simplified real-life situation (i.e., car moving on a frictionless road), and then “play and experiment with” their models to explore more about the different components of the phenomenon (e.g., force, mass, velocity, acceleration). However, it was evident that the interactive component of learning went deeper than just a surface level manipulation of the models, as the generative dimension of model construction was clearly present during the intervention (Schwarz et al., 2009; Wilkerson-Jerde et al., 2015). A thorough review of the classroom observation notes showed that there was a rich discussion in the classroom fostering a holistic conceptual understanding about force and motion concepts while students were building their computational models. The observation notes indicated that most of the students in each class period talked to their peers sitting next to them during the programming and model building activities. They asked questions to each other regarding the science concepts, used their model to articulate and convey their understanding of the science concepts to their peers, and discussed if their model needed any revision or modifications. Some of the questions that students had to answer to build their models included ‘What is the relationship between the mass and acceleration of an object if there is a constant force being applied?’, ‘How to find the velocity of the car if its acceleration is known?’, ‘What the behavior of the car should be if the force
acting on it becomes zero while it is already moving?’. It was the generative dimension of constructing models where students were actively engaged in defining and building the “rules” of the target phenomenon in terms of algorithmic steps with programming blocks, and then experimenting with their models to examine how the physical phenomenon works as well as using their models as a shareable external artifact of their state of understanding to discuss and communicate their ideas with their peers and teacher. It is unlikely that just working with a pre-built computational model would provide similar discursive opportunities in the classroom. 

Computational modeling activities enabled students to explore force and motion concepts through creating and working with multiple representational systems including mathematical representations of physical laws in the form of block-based programming code and dynamic visual representations of physical phenomena in the form of graphical real-world objects. These two representations are inherently linked to each other as the behavior of the visual output of the program is governed by the block-based code that the model is built on. Each representation has its own specific affordances and constraints. For example, Figure 5.1 shows the mathematical formula relating to Newton's Second Law of Motion and the block-based code in Scratch calculating the acceleration using the equation. In this example, student created a variable for each element on the equation including the force applied on the object, the mass of the object and the acceleration of the object. Then, using the set block, student calculated acceleration by dividing the applied force to the mass of the object.
While the above example shows a re-generation of a mathematical equation in terms of visual code, block-based programming also served as a medium to help students contextualize highly abstract mathematical relationships behind the physical laws (Sherin, diSessa, & Hammer, 1993, Sherin, 2001; Wagh, Cook-Whitt, & Wilensky, 2017). For example, Figure 5.2 shows how velocity of an object can be calculated using the acceleration of the object. Here, students did not re-generate the equation to find the velocity. Instead, they applied their conceptual understanding to contextualize the relationship between velocity and acceleration. Since acceleration is the rate of change of velocity per unit time, the magnitude of velocity can be calculated in real-time by adding to its value the magnitude of acceleration at every second. In other words, if an object is moving to the west with an acceleration of ‘five’ meters per second square, then its velocity to the west will increase by ‘five’ meters per second at every second. The change block simply adds the value of its parameter (i.e., acceleration) to the actual value of the velocity variable. Since it is inside a forever block, its value will be calculated in real-time based on the value of acceleration and the velocity of the object will change accordingly when the model runs. So, instead of directly re-generating the equation in block-based code to find velocity from acceleration,
students had to apply their conceptual understanding of the relationship between velocity and acceleration to construct an algorithm for calculating velocity in this context of a car on a frictionless surface.

![Figure 5.2. Calculating velocity using block-based code](image)

Lastly, block-based programming medium allowed students to turn the mathematical abstractions (i.e., equations) into *causal relationships* in terms of programming instructions in the form of pre-built programming blocks (diSessa, 2000). For example, the equation Force = mass x acceleration (F = m\*a) does not actually tell anything about the direction of causation (i.e., is it force applied on an object that cause an acceleration or the other way around?) (Sherin, diSessa, & Hammer, 1993; Sherin, 2001). During the model building activities, it was the students’ responsibility to figure out causal relationships among the variables, which is an impetus for conceptual change, while re-creating this mathematical abstraction in a different representational system (e.g., Figure 5.2). Using block-based programming for formulating an algorithm to represent a physical phenomenon (e.g., the effect of a force on the motion of a car on frictionless road) required students to think about
which variable causes a change in another variable, which is not usually obvious in the algebraic notation of the laws governing the behavior of physical phenomena. As shown in Figure 5.2, students had to determine the causal relation between force and acceleration, and between velocity and acceleration. In other words, they had to figure out that it is the force that causes a change in the acceleration of an object and it is the acceleration that causes a change in the velocity of an object. It is also evident from the algorithm that no net force means zero acceleration, and zero acceleration means no change in velocity.

The researcher argues that employing block-based programming to construct simulation-based models has unique affordances. First, block-based programming makes it significantly easy for students without any prior programming experience to engage in computational thinking concepts and practices in science classrooms. Secondly, constructing computational models using block-based programming requires students to think and formulate the rules (science concepts) that govern the behavior of a physical phenomenon. Lastly, the dynamic visual output of the simulation allows students to actually test and observe how their code (i.e., algorithm) behaves in real time. Is it possible to determine how much working with each representational system (i.e., constructing block-based code, experimenting with visual output) contributed to the participating students’ significant learning gains? The collected data does not allow the researcher to draw an inference on the magnitude of the effect of working with each representation on student learning. Further research is needed to investigate how working with each representation system in computational modeling environments that are similar to the one presented in this study contributes to conceptual understanding of science concepts in formal classroom settings.
Research Question 2. How does participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment influence middle school students’ understanding of computational thinking concepts and practices?

a. Is there a differential impact of the classroom intervention on students’ understanding of computational thinking concepts and practices according to students’ gender?

One of the purposes of the intervention of this study was to introduce students to computational thinking concepts and practices through Scratch block-based programming environment. The Computational Thinking Test (CTt; Roman-Gonzalez et al., 2017) was used as a pre- and post-test to assess students’ understanding of computational thinking concepts and practices prior to and after the classroom intervention. Students’ responses to the reflective statement question at the end of each class period as well as their responses to the interview questions after the classroom intervention provided additional insights about their understanding of computational thinking concepts and practices.

The results of the pre-test (Table 4.10) show that prior to the classroom intervention students’ knowledge about computational thinking concepts and practices were moderate ($M = 58.37, SD = 12.77$; out of 100 on Rasch scale). The histogram of frequencies of students’ pre-test scores on Rasch scale shows that majority of the students’ responses were clustered between 40 and 80 with 63 as the mode of the distribution (Figure 4.3), which indicates that most of the participating students had some experience in block-based programming. In fact,
many students stated during the classroom intervention that they attended one or more Hour of Code activities in the previous years, and some students were also enrolled in an after-school robotics club, which can explain participating students’ overall modest performance on the pre-test.

This study also aimed to advance students’ knowledge of computational thinking concepts and practices by engaging them in a week-long programming and model building activities. The results of the paired-samples t-test comparing students’ pre- and post-test scores on the CTt (Table 4.11) showed a statistically significant increase in students’ understanding of computational thinking concepts and practices at the end of the classroom intervention with a large effect size of 0.91. Students’ mean score for the CTt on the post-test ($M = 65.33, SD = 11.05$; out of 100 on Rasch scale) increased by 6.96 logits compared to their mean score on the pre-test ($M = 58.37, SD = 12.77$; out of 100 on Rasch scale). Although there were significant learning gains with a large effect size as a result of participating in the intervention, students did not demonstrate a particularly high level of understanding of computational thinking concepts and practices in the post-test, with their average post-test score on Rasch scale was $M = 65.33 (SD = 11.05)$. The histogram of frequencies of the post-test scores shows that majority of the students’ responses were clustered between 50 and 80 with 66 as the mode of the distribution (Figure 4.4). For most of the students, it was their first time to be formally exposed to computational thinking concepts and practices in classroom settings. Therefore, it would not be reasonable to expect that students would demonstrate a high level of understanding of computational thinking concepts and practices just by attending a five-day intervention. However, the significant learning
gains on computational thinking concepts and practices during the intervention indicate that students can achieve higher levels of understanding if computational thinking is explicitly integrated into science instruction throughout the academic year. According to the classroom observation notes and interview responses, several students expressed that they wanted to spend more time freely practicing the computational concepts themselves using Scratch programming environment. Due to time constraints, it was not possible to do this during the intervention. However, the researcher recommends allocating more time to students for self-practice and self-exploration during programming exercises, so that students can do repeated practices in different contexts (e.g., storytelling, game building) that they can choose, and reinforce their understanding of computational concepts. It is important to note that the researcher spent three days just on programming to make sure that every student had the same base programming ability, and this time was taken away from science related learning activities. However, the relatively high prior knowledge of computational thinking meant that the learning in the area of computational thinking was modest. Future studies conducting longer term interventions can assess students’ prior knowledge of computational thinking, and then adjust the programming and computational thinking activities to differentiate the instruction and allocate more time across learning activities related to science.

The study also investigated whether there was a differential impact of the classroom intervention on students’ understanding of computational thinking concepts and practices based on gender. Previous research (Duncan & Bell, 2015; Ericson & McKlin, 2012; Margolis et al., 2000; Orton et al., 2016) suggested that gender may be a significant predictor of students’ confidence and self-efficacy towards learning about computational thinking.
concepts, which in turn may affect their learning gains in the classroom. A paired-samples t-test was run to compare both male and female students’ performance on the CTt prior to and after the classroom intervention. The results of the paired-samples t-test (Table 4.12) showed that both male and female students’ average scores on the post-test increased significantly compared to the pre-test. However, the results also showed that female students had a lower mean score on both pre- and post-test and had higher learning gains compared to male students with a three times larger effect size (Table 4.12). The results of the independent-samples t-test comparing male and female students’ computational thinking learning gains showed that there was a statistically significant difference in the learning based on gender where female students demonstrated higher learning gains on computational thinking concepts (Table 4.13). This indicates that female students had less experience with computational thinking compared to male students before the intervention but showed greater capacity to make gains when provided the opportunity. Overall, these findings are in agreement with the findings of the previous studies in the literature showing significant learning gains for both male and female students on computational thinking assessments (McDowell, Werner, Bullock, & Fernald, 2003; Werner, Denner, Campe, & Kawamoto, 2012), even in the case where female students expressed less interest and self-efficacy towards using computers and computational tools (Orton et al., 2016). For example, in their study with a large sample of middle school students (N = 311), Werner and colleagues (2012) used the Alice visual programming environment to teach computational thinking in an after-school class and developed a performance assessment based on students’ work in the
learning environment. Using correlation analysis, the researchers found that students’ total scores on the assessment were not related to their gender.

In addition to the pre- and post-test results, students’ responses to the reflective statement questions as well as their responses to the interview questions provided in-depth insights about their understanding of computational thinking concepts and practices. Here is an example of a student’s response to the reflective statement question of “What did I learn today?” showing her understanding of conditionals and decision-making in computer programming: “that to take out many options or commands, then you would use the if then, else block”. By saying “many options”, student was probably referring to the different conditions she coded in her programs to make the sprite take different decisions based on the given conditions using if and if-else statements. The concept of making decisions to take certain actions based on different conditions is an important computational concept in programming, and it is often taught in introductory programming courses (K-12 CS Framework, 2016). Here is a reflective statement response of another student that indicated his understanding of conditionals: “Today I learned how to make the cat make a decision with many questions”.

Another important computational concept in programming is loops and repetition/iteration structures (Brennan & Resnick, 2012). In programming, different looping statements are used for running a particular set of instruction multiple times during the execution of the code. Previous research suggested that the concept of looping in programming is difficult for students to master (Robins, Rountree, & Rountree, 2003; Grover, Pea & Cooper, 2015; Zur-Bargury et al., 2013). There are different looping
statements in Scratch including *forever, repeat*, and *repeat until*. *Forever* keeps executing the instruction inside it until the program stops running. *Repeat* runs the instruction inside it for a certain number of times that user specifies. *Repeat until* is similar to *repeat* except that it runs the instruction inside it until the condition in its parameter becomes false. Here is a student’s response to the reflective statement question showing evidence of her understanding of looping in programming: “I learned how to use loops and conditions to run separate pieces of code over and over”. This student articulated her understanding of looping by stating that loops execute a certain set of instructions (“separate pieces of code”) multiple times (“over and over”). Another student’s reflective statement response shows evidence of his understanding of *repeat* and *repeat until* looping structures: “I learned how to make something do something for some amount of time and until certain conditions are met”. The student described the *repeat* loop by saying that it runs “somethings for some amount of time” where time is entered in seconds as the parameter of the loop statement. The student also described the *repeat until* loop by saying that it executes an instruction “until certain conditions are met”. Here is another reflective statement response that a student elaborated on the *repeat* loop: “I learned about loops like repeating blocks that repeat the script inside that block how many ever times it wants”.

The researcher identified several challenges faced by the participating students during the implementation of the programming and model construction activities in the classroom using the classroom observation notes and students’ interview responses. Firstly, the majority of the participating students struggled outlining and formulating an algorithm when they were given a relatively difficult programming task such as building a pong game. Devising
an algorithm requires examining the target problem and breaking it down into a number of smaller problems (i.e., problem decomposition), and then formulating steps to solve each problem, one at a time. Algorithms can be composed in several ways including writing the solution steps in a natural language like English, drawing flowcharts, or writing a pseudocode (K-12 CS Framework, 2016). The next phase includes translating the algorithm into code using the computational constructs such as conditionals, loops, and variables.

During the learning activities, the researcher repeatedly explained as well as modeled the processes involved in formulating an algorithm to solve computational problems by writing sample steps on the whiteboard as well as showing flowcharts on the PowerPoint presentation. After giving students a programming task, such as building a pong game, the researcher emphasized that they should start by first carefully thinking about the problem and devise an algorithm. The researcher suggested students take notes regarding the design of their algorithms on a piece of paper or using a word processor application such as Microsoft Office Word on their computers. However, the classroom observation notes indicate that majority of the students skipped this step and directly delved into coding—attempting to formulate an algorithm while they were coding. This was problematic as some students struggled to formulate an algorithmic solution while navigating through the programming environment at the same time. This suggests that instructor’s modeling of the algorithm generation processes verbally in the classroom with the aid of flowcharts may not an effective scaffolding for students to engage in formulating algorithmic solutions before starting coding. A better approach may include explicitly teaching students about writing pseudocode and translating it into code using computational constructs. A pseudocode is
basically a high-level description of an algorithmic solution that highlights the "flow" of the solution and can be generated using simple sentences and programming constructs (e.g., IF this condition is TRUE, THEN execute this action). Figure 5.3 shows an example pseudocode for a game algorithm that was adapted from Davies (2008). Several researchers recommended engaging students in creating pseudocodes to draft an algorithmic solution to the given problem during introductory programming exercises (Davies, 2008; Futschek, 2006; Grover, Cooper, & Pea, 2014; Haden, Gasson, Wood, & Parsons, 2016).

```
1. Make PLAYER NUM hold 1.
2. Ask the user, "How many at bats for #PLAYER NUM?"
3. Remember what the user says and call it AT BATS.
4. Make OUTF AT BATS hold 0.
5. If AT BATS is not -1, do these steps, then check it again:
   a. Ask the user, "How many hits for player #PLAYER NUM?"
   b. Remember what the user says and call it HITS.
   c. Ask the user, "What is player #PLAYER NUM's position?"
   d. Remember what the user says and call it POSITION.
   e. If POSITION is either "leftfield", "centerfield", or "rightfield", do these steps:
      i. Make OUTF AT BATS hold OUTF AT BATS plus AT BATS.
      ii. Make OUTF HITS hold OUTF HITS plus HITS.
```

Figure 5.3. A sample pseudocode for a game algorithm (adapted from Davies, 2008)

Another challenge faced by the participating students was that when their program did not run as expected, they were not able to easily diagnose the errors (i.e., unexpected outcomes that keep a program from working correctly, and commonly called as "bugs") within their code because Scratch does not have a built-in debugging feature. Casey (1997) defined debugging as "the process of locating and correcting errors within a program" (p.
Debugging is a crucial skill in programming and requires complex reasoning (Grover, 2016). The classroom intervention of this study did not explicitly teach students about debugging. However, the researcher explained to students that when their code did not work as expected, they could try to locate the error(s) by dismantling the programming blocks in their scripts, running each block or groups of blocks successively, and observing the output of the program to isolate the error in the code. The researcher showed a sample debugging process step-by-step in Scratch a number of times on the screen. However, students seemed to have struggled doing this on their own. Some advance block-based programming environments such as ViMAP (Sengupta et al., 2012) has a relatively intuitive built-in debugging feature, where students can run the program line by line in real-time, and start and stop at any time while observing the output, which can make it easier for students to isolate the error in the code. However, there is a higher learning curve for using more advanced block-based learning environments such as ViMAP, and the instructor needs to make an informed decision based on the available time allocated for these activities. If the computational thinking practices are planned to be integrated in the context of computational modeling in a science classroom throughout the semester, then the teacher may want to prefer using a more advanced block-based programming environment that has a built-in debugging feature.
Research Question 3. How do middle school students’ attitudes towards programming change as a result of participating in a one-week intervention course introducing computational thinking practices and simulation-based model building through a visual block-based programming environment?

a. Is there a differential impact of the classroom intervention on students’ attitudes towards programming according to students’ gender?

Lastly, this study aimed to investigate how middle school students’ attitudes towards programming changed as a result of attending a week-long classroom intervention where they actively engaged in learning about programming and constructing computational models of physical phenomena using the Scratch visual block-based programming environment. An eight-item, five-point attitudinal survey (1: Strongly Disagree, 5: Strongly Agree; see Table 4.16 for individual survey items), which was adapted from the Science subscale of the S-STEM survey (Unfried et al., 2015), was used as a pre- and post-test to assess students’ attitudes towards programming before and after the classroom intervention. In addition, several qualitative data sources including students’ interview and reflective statement responses, and the classroom observation notes were analyzed to provide a better understanding of the participating students’ attitudes towards programming.

The results of the attitudinal survey on the pre-test (Table 4.15) indicate that prior to the classroom intervention participating students, on average, had neither favorable attitudes nor unfavorable attitudes towards programming with their aggregated mean score was very close to 50 on Rasch scale ($M = 49.37$, $SD = 13.29$; out of 100 on Rasch scale). The
histogram of the students’ pre-test scores showed that a large number of students had a score of 50 and majority of student’s scores were distributed around 50 (Figure 4.5). Since majority of the students stated that they had participated in one or more Hour of Code activities in the past (confirmed by the classroom teacher), the researcher expected that, on average, students would have relatively positive attitudes towards programming as the primary purpose of the Hour of Code activities is to get students excited about programming by showing them computer science is fun and creative (Partovi, 2015). One possible explanation could be that students were not given age-appropriate challenges during the Hour of Code activities as there are a lot of programming challenges available online with different levels of difficulty based on the age of the participant. Teachers who conduct Hour of Code are supposed to go through the challenges on their own and choose the most appropriate one(s) for their students to play based on their age and academic levels. Giving students some random challenges that are too easy or too difficult may potentially make them be less interested in programming, now or in the future (Sharek & Wiebe, 2015). Further research exploring how and to what degree Hour of Code activities influence students’ attitudes towards programming and computer science is needed since for most students it is the only time that they are exposed to computational concepts in the school.

To investigate how middle school students’ attitudes towards programming change as a result of attending the week-long classroom intervention of this study, a paired-samples t-test was conducted comparing students’ pre- and post-test scores on the attitudinal survey. The results (Table 4.17) show a statistically significant increase in students’ scores on the post-test with a moderate effect size ($t(81) = 5.93, p < .001, d = 0.66$), indicating students
had more favorable attitudes towards programming on the post-test compared to the pre-test. While this specific CT attitudinal survey was not rigorously analyzed with regards to reliability and validity, it is based on the S-STEM instrument, which has undergone such rigorous evaluation (Unfried et al., 2015).

There is only a limited number of studies in the literature that investigated the change in students’ attitudes towards programming or computer science as a result of a treatment, and the results of this study are in agreement with the findings of the previous studies showing a positive shift in students’ attitudes (Ericson & McKlin, 2012; Orton et al., 2016; Wolz et al., 2011). For instance, in their study with middle and high school students, Ericson and McKlin (2012) investigated the effect of an eight-week computing summer camp on students’ attitudes towards programming. During the summer camp, students were introduced to block-based programming and robotics, using a variety of programming environments and tools including Scratch, App Inventor, and WeDo robotics kits. The researchers used pre- and post-surveys to assess participating students’ attitudes towards programming, which included five-point Likert-scale items such as “Programming is hard”, “I am good at programming”, and “Computers are fun” (p. 291). The results showed statistically significant positive changes in students’ attitudes towards programming (Ericson & McKlin, 2012).

In addition to the pre- and post-test results, several qualitative data sources including the classroom observation notes, students’ responses to the reflective statement questions and the interview prompts provided deeper insights about their attitudes towards programming as well as computational modeling. Overall, the classroom observation notes suggested that
majority of the participating students seemed to have enjoyed the programming and modeling activities during the intervention. There was a small number of students in each classroom who did not show much interest or did not want to engage in the programming activities or even tried to disrupt other students, however the school teacher stated that those students were behaving similarly in other classes too. Nonetheless, most of the students showed excitement about their learning of computational concepts and modeling during the intervention and actively engaged in the learning activities.

All students who participated in the interviews \((n = 12)\) after the intervention showed very positive attitudes towards learning about programming (Table 4.20). When asked about whether they enjoyed learning computer programming, one female student stated: “I enjoyed a lot. I didn’t know programming before, so it was fun to learn how you can put different blocks on things to see how you can make other things maybe move or talk or dance”. It is important to point out here that participation to the interviews was completely voluntary and it is possible that only the students who already had positive attitudes towards programming volunteered for the interviews. Moreover, three of the interviewees were very quiet students in the classroom during the intervention, and mostly engaged in the activities by following the instructions. However, during the interviews these students expressed their enjoyment of the learning activities. It is possible that self-selection bias influenced the overall positive reporting in the interviews.

Students’ responses to the reflective statement question at the end of each class period during the intervention also provided additional insights about their attitudes towards programming and the learning activities in general. In their responses to the question of
“What did I learn today?”, a lot of students provided affective responses (Table 4.21). Here is one student’s response from the first day of the intervention: “Today I learned the basics of how to code using scratch command blocks. I also learned how to make a simple game of ping pong. The lesson was really fun and easy to learn.”. Another student responded by saying: “i learned that you can make any game by just putting a little of steps and it's really cool”. While majority of the students provided responses that showed evidence of their positive attitudes, a small number of students’ \((n = 4)\) responses indicated negative attitudes. For instance, one student said: “i learned that coding is very hard so i don’t really like it. But i guess if it was something i had to do”. Along the same lines, another student said: “nothing because it was very difficult and I didn't [sic] really like”. While these two students reflected that the programming activities were difficult for them, one student found them too easy: “I’m glad this is the last day because this class was so boring. These blocks aren’t programming. Teach me python or Java script”. This student probably had a strong background in programming and the introductory block-based programming activities did not teach him anything new. Since there is a push to introduce computing to students of all ages, and more students will probably be accessing online high-quality introductory programming materials through websites such as Code.org for free, it is important that teachers who plan to integrate computational thinking concepts into their curricula need to formulate strategies to differentiate the instruction to meet the needs of all students.

The present study also aimed to investigate whether there was a differential impact of the classroom intervention on students’ attitudes towards programming based on gender, since previous studies in the literature (Duncan & Bell, 2015; Ericson & McKlin, 2012;
Margolis et al., 2000; Orton et al., 2016) suggested that gender has a significant role in shaping students’ attitudes towards programming and computer science. The results of paired-samples t-tests (Table 4.18) comparing separately both male and female students’ pre- and post-test scores on the attitudinal survey showed that participating in the classroom intervention of this study resulted in a statistically significant increase in both male and female students’ attitudes towards programming, indicating no differential impact of the intervention on students’ attitudes based on gender. While both male and female students’ attitudes towards programming became more favorable at the end of the intervention, the researcher also compared the change in attitudes between male and female students by running an independent-samples t-test to find if the change in pre and post scores on the attitudinal survey differed significantly between the two gender groups. According to the results (Table 4.19), female students on average had a higher change in their pre to post scores on the attitudinal survey compared to male students, but the difference between the two groups was not statistically significant. It is also important to note that female students had lower average scores on attitudinal survey compared to male students both before and after the classroom intervention. Lastly, the classroom observation notes showed no discrepancy between male and female students in terms of their attitudes towards programming as well as engagement in the learning activities.

Overall, the results of this study suggest that participating students’ attitudes towards computer programming significantly changed in the positive direction, and the intervention did not have a differential impact on students’ attitudes based on gender. It is important to examine the factors that made students gain more favorable attitudes towards programming
in a week-long classroom intervention. In other words, what specifically about the intervention that helped students have more positive attitudes towards programming? During the interviews, almost all participating students stated that they liked programming and enjoyed the activities very much because it was “hands-on”, “fun”, and “engaging”. Here is a student’s response who thought the programming activities were fun: “I really enjoyed learning programming, it was fun like to put the blocks and see how it works. You can play around with it and see what works better and then if you have any mistakes you can do better in future.” Another student response shows how programming can help students express their creativity:

I enjoyed learning computer programming because now I can make the characters actually do things what I want them to and make things move. I can now actually make those things and change in any way I want because I know programming. And what I liked most was that I can actually make the characters talk you know with bubbles. It was pretty fun.

Students’ responses regarding the modeling activities were also similar as most of them emphasized the hands-on nature of the activities:

I enjoyed doing it for the Newton’s laws of motion. It helped me get a better understanding of like some of the formulas we used because when we did it, it was all word problems and just the formula on paper. But in here we had the simulation and we were able to participate in it and it was hands-on.

Another student wanted that this kind of modeling activities should be part of the normal science curriculum:

Okay, I would like to definitely do this more in science because I find it different I mean a better way to learn than the usual like slideshow pack and fill in answers. It is not hands-on. I feel like I learn better hands-on because you have to think about it
yourself, you can’t be like sleeping putting your head down. I really like when it is hands-on you know. And I would have definitely enjoyed science classes a lot more this year if this had been an actual part of the curriculum like we do projects on our own and we are graded on them you know.

It is evident from the results that integrating computational thinking concepts into science instruction in the context of model building activities in the classroom can make science lessons both more attractive and more authentic to students, and this promotes conceptual science learning as the results of this study showed. There is, of course, the additional benefit of enhancing computational thinking abilities while also providing engaging, efficacious science learning opportunities.

**Limitations of the Study**

There are several limitations of this study. First, the sample of this study involved a group of seventh-grade students in one public middle school in the southeastern part of the United States. The sample of this study may not be representative of the seventh-grade student population in the United States. This may limit the generalizability of the findings of this study to larger student population.

Secondly, the student interview data may be biased since the interviews were volunteer-based (i.e., self-selecting) and most of the volunteering students seemed to be very interested in block-based programming during the intervention and actively engaged in the learning activities. Therefore, their responses especially regarding their attitudes towards computational thinking and programming may not necessarily be representative of the larger group of students who participated in this research. Also, it is important to note that not all students who engaged in the learning activities of the classroom intervention consented to
participate in the research, which might have biased the overall results presented in this study. Third, there were several English Language Learner (ELL) students, mostly from Hispanic origins, in each participating classroom of this study and the classroom observation notes showed that those students sometimes struggled to follow the instructions, which might have potentially prevented them from fully participating in the learning activities. For example, the programming blocks in Scratch use simple English words such as forever, repeat until, set, change, glide and so on. A student whose native language is English can easily understand the difference between the looping structures forever and repeat until without even explicitly explaining it during the instruction. However, this may not be the case for an ELL student. In fact, some students were using the Google Translate website during the activities to translate some of these words into Spanish so that they could make sense of the target blocks that they wanted to use in the programs. While the researcher tried to provide scaffolding for the ELL students by explaining them important concepts and terminology using a simpler language, those students might not have benefited from the learning activities as much as other students whose native tongue are English.

This study employed an immediate post-test to assess student learning during the classroom intervention. However, this approach does not provide any evidence regarding the lasting effects of the intervention on student learning. Future research employing both immediate and delayed post-tests is needed to assess long-term effects of similar classroom interventions on students’ understanding of science and computational thinking concepts.

The Computational Thinking Test (CTt), which was used to assess participating students’ understanding of computational thinking concepts and practices prior to and after
the classroom intervention, entirely consisted of multiple choice questions. The developers of the test, Roman-Gonzalez et al. (2017), pointed out that the CTt “might be measuring CT [computational thinking] at its lower cognitive complexity levels (‘recognize’ and ‘understand’) (p. 688). Therefore, this study might not have assessed students’ computational thinking concepts and practices at higher levels of complexity. Lastly, the learning activities were not differentiated based on students’ prior experiences in block-based programming, as measured using the CTt before the intervention. Overall, there were a few students who explicitly expressed their boredom during the learning activities, especially in the first three days of programming activities, in their reflective statement responses. It is possible that those students along with others who had extensive experience in block-based or traditional programming languages did not benefit much from the learning activities compared to students with minimal or no programming background.

Lastly, this study did not employ a control group that could be used to compare the results from the experimental group and determine if the learning and attitudinal gains were due to the classroom intervention alone.

Conclusion and Future Work

Computational thinking and modeling are authentic practices that scientists and engineers use frequently in their daily work (NRC, 2012). Advances in modern computing technologies over the past couple of decades have further emphasized the centrality of modeling in science by making computationally-enabled model use and construction more accessible to scientists today. Therefore, it is crucial for middle and high school students to get exposed to these practices in their science classrooms. The findings of this study indicate
that infusing computational thinking concepts and practices into science instruction in the context of simulation-based model building activities can further students’ conceptual understanding of science concepts as well as computational thinking concepts and practices. The results also show that participating students had more favorable attitudes towards programming at the end of the intervention. The findings of this study suggest that even with minimal time in the classroom to introduce computational thinking concepts and practices through block-based programming can enable students to build relatively advanced computational models, which in turn promote conceptual learning in science classrooms. This study contributes to the evolving literature on integrating computational modeling into science curricula by emphasizing the affordances and generative dimension of model construction through block-based programming.

It would most likely be necessary to provide students similar opportunities on a regular basis if we would want this positive affective and cognitive shift to be sustained. The intervention of this study lasted for five class periods (50 min each period) in total for each participating classroom. Future research should investigate how integrating computational thinking concepts and practices into grade level science curricula over longer time frames such as a semester can influence conceptual science learning for other science topics (e.g., life sciences).

Since classrooms are getting more diverse with increasing number of ELL students, future studies should explore what kind of scaffolds can be provided by teacher and the by programming environment to these students during programming and modeling activities,
which can help them overcome language barriers and focus on science content. This can be particularly helpful for students whose native tongue is English but have reading difficulties.

In this study, we did not extensively explore how employing computational thinking practices through a model-based pedagogy affects classroom discourse. The classroom observation notes indicated that there was a rich discussion among students about force and motion concepts during the model building activities. Future research investigating how computationally-enabled model-based pedagogy shapes classroom discourse is needed to assess how this classroom discourse helps students learn science and computational thinking concepts.

The intervention was conducted by the researcher of this study. Future research should investigate prospective and in-service teachers’ self-efficacy towards using computational tools and modeling pedagogy in the classroom. If this approach of integrating computational thinking concepts and practices into science instruction in the context of model building activities is also found successful by other studies, then it will be the science teachers who will be given the responsibility to employ this pedagogy in the classroom. Therefore, it is important to study teachers’ self-efficacy as well as attitudes towards programming and computational thinking and how their self-efficacy and attitudes influence their classroom practices when they are implementing model-based pedagogy.
REFERENCES


http://dx.doi.org/10.1016/S0160-2896(00)00050-7.


and Sons.


consequences of inquiry-based Logo instruction. Journal of Educational Psychology,

80(4), 543-553.

Linacre, J. M. (2017). Winsteps Rasch measurement computer program. Beaverton, Oregon:

Winsteps.com.


thinking through programming: What is next for K-12?. Computers in Human Behavior,

41, 51-61.


programming language and environment. ACM Transactions on Computing Education


Appendix A

Parent Consent Form

North Carolina State University
Informed Consent Form for Research
Title of Study: Exploring Computational Thinking in Middle School Science Classrooms
Principal Investigator: Dr. Eric Wiebe (NC State University)

We are asking your child to participate in a research study. The purpose of this study is to examine how participating in a one-week activity that introduces computational thinking practices and computer modeling will support students’ computational thinking practices and conceptual knowledge of Force and Motion.

Information
If you allow your child to participate in the study, their recorded data resulting from the learning activities during the course facilitated by the researcher would be collected and used for research purposes only. The recorded data resulting from the learning activities will include pre/post test data, short written responses to reflective statement questions, programming and modeling artifacts, and audio recordings of student interviews. Pretest and posttest are identical, consist of two multiple-choice assessments, and an attitudinal survey. Reflective statement question is “What did I learn today?” and students will provide a short response to this question at the end of each class period. Students will respond to pretest, posttest, and reflective statement questions online using the Qualtrics software. Students’ programming and modeling artifacts will be digital files of computer programs that they will build using visual programming environment. Students’ audio recordings will be transcribed by removing any identifying information. Then the audio recordings will be destroyed immediately.

The activities will take place in science classrooms during normal class hours. At the beginning of the activity, students will take a test on computational thinking skills, a test on Force and Motion, and a short survey on attitudes toward computer programming. Then, students will engage in learning activities on computational thinking and computer modeling. Student will produce several programming and modeling examples while engaging in the learning activities. At the end, students will take the same tests and survey, so the researcher can assess the effectiveness of the activity by comparing students’ pre-post test scores and survey responses.
Pre- and post-tests are for research purposes only and will not affect students’ class grade in any way. At the end, students can volunteer to participate in an interview where the researcher will ask some questions regarding students’ understanding of computational thinking practices as well as science content. Students participating in the interview will be given a $10 Target gift card for compensation. The interviews will be audio recorded by the researcher.

Participation in this study is not a requirement of your child’s science class. Student participation, or lack of participation, will not affect student’s class grade in any way. If the student chooses not to participate in the study or the parent does not give permission, student will be doing the same activities, but the information from those activities will not be used for research purposes. The learning activities students will engage in this study will be part of their normal school coursework, and they will complete the activities regardless of their participation in the study.

**Risks**
Participating in this study involves minimal risks associated with regular computer usage.

**Benefits**
If you allow your child to participate, you will contribute to the advancement of research on how students can learn about computational thinking practices in the context of computer modeling in middle school science classrooms.

**Confidentiality**
Consenting students will be given a unique identifier that will be randomly generated and logged with the student's data, so student's name or identity will not be associated with any data that will be collected during the research. No reference will be made in oral or written reports which could link your child’s name or identity to the study. No photographs or video images of students will be taken during the research. A password-protected cross reference Excel document that includes students’ names and assigned unique IDs will be stored in the researcher’s two-factor authenticated university-owned Google Drive during the course of the data collection. Once the data collection is finished, the researcher will destroy the password-protected cross-reference document on the Google Drive immediately, so that no one can match your child’s identity to the data that will be analyzed and possibly published. If a student participates in an interview, the audio recording will be transcribed, removing any identifying information, and the audio recording destroyed.
Contact
If you have questions at any time about the study or the procedures, you may contact Dr. Eric Wiebe, principal investigator, at 919-515-1753. If you feel you have not been treated according to the descriptions in this form, or your rights as a participant in research have been violated during the course of this project, you may contact Deb Paxton, Regulatory Compliance Administrator, Box 7514, NCSU Campus (919-515-4514).

Participation
Your child’s participation in this study is voluntary. You may decline participation without penalty. If you decide to participate, you or your child may withdraw from the study at any time without penalty.

Consent
“I have read and understand the above information. My child has permission to participate in this study with the understanding that they may withdraw at any time. My child can participate in this research and the data collected during the research can be used by the researcher for educational research purposes.”

Student’s Name: ___________________ Parent/Guardian’s Name: ___________________

By signing my name below, I certify that I agree with the above statement:

Parent/Guardian’s Signature: ___________________ Date: ________________
Appendix B

Student Assent Form

North Carolina State University
Student Informed Assent Form for Research
Title of Study: Exploring Computational Thinking in Middle School Science Classrooms
Principal Investigator: Dr. Eric Wiebe (NC State University)

Dear Student,

You are invited to participate in a study that aims to help you learn about computational thinking and modeling. Computational thinking is an increasingly important skill in all science disciplines. Learning about computational thinking with modeling will help you understand scientific concepts better.

What will I do in this project?
This study will take place in your regular science classroom and take seven class periods in total. In the first period, you will answer questions that assess your knowledge of computational thinking, and Force and Motion. You will then respond to a short survey about your attitudes toward programming.

In the second period, you will start to learn about computational thinking. You will engage in programming with Scratch and write computer programs. After you learn programming, you will engage in model building activities to improve your science knowledge.

At the end, you will answer the same questions. This will allow researcher to evaluate the effectiveness of the course. The questions you will answer are for research purposes only and will not affect your class grade in any way.

We plan to do interviews with volunteer students at the end of the study. If you volunteer for interview, you will receive a $10 Target gift card.

After participating in this study, you will be able to write computer programs in Scratch and build computer models.

Some of the data sources that we will collect will include pre/post test data, short written responses to reflective statement questions, programming and modeling artifacts, and audio recordings of interviews. Pretest and posttest are identical, consist of two multiple choice assessments, and an attitudinal survey. Reflective statement question is “What did I learn today?” and you will provide a short response to this question at the end of each class period.
You will respond to pretest, posttest, and reflective statement questions online. Programming and modeling artifacts will be digital files of computer programs that you will build using visual programming. Audio recordings of interviews are audio files that will be transcribed by removing any identifying information. Then the audio recordings will be destroyed immediately.

Participating in this study involves minimal risks associated with regular computer usage. If you choose not to participate or do not have parent permission, you will still do the same activities but the information from those activities will not be used for research purposes. The learning activities you will engage in this study will be part of your normal school coursework, and you will complete the activities regardless of your participation in the study.

**What if I have questions?**
Your participation is voluntary. You have the right to stop participating in this study at any time and for any reason. You can also ask questions at any time. If you have any questions about this study, you may contact Dr. Eric Wiebe, principal investigator, at 919-515-1753. You may also contact Deb Paxton, Regulatory Compliance Administrator, Box 7514, NCSU Campus (919-515-4514) if questions or problems arise during the study.

If you agree to participate in this study, please sign your name below. Thank you!

___________________________  ___________________________  ____________
Student Name                  Student Signature              Date
Appendix C

Computer Programming Attitude Scale

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neither Agree nor Disagree</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I am sure of myself when I do programming.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2. I would consider a career in programming.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>3. I expect to use programming when I get out of school.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>4. Knowing programming will help me earn a living.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5. I will need programming for my future work.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>6. I know I can do well in programming.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>7. Programming will be important to me in my life’s work.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8. I am sure I could do advanced work in programming.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
Appendix D

Script to Recruit Students to Volunteer for the Interview

Dear students,
Thank you for taking part in this study. We greatly appreciate your cooperation. As I mentioned earlier, participation in this study will not affect your class grade in any way. I hope you enjoyed the learning activities and have a better understanding of programming and modeling. As the last part of this study, I would like to interview individually with ten volunteer students. The participation to the interview is voluntary and will take about half an hour. In the interview, I will ask you questions about the models you built over the last couple of days.
The purpose of these interviews is to better understand how the activities helped you learn about Force and Motion. I will audio record the interviews and transcribe them later for data analysis. No information about your identity will be recorded during the interviews or associated later. The interviews will take place in the school library either during the lunch time (after you finish your lunch), or after school depending on your availability.
If you choose to volunteer for interview, you will receive a $10 Target gift card as a compensation. If more than two students choose to volunteer from the same class, I will randomly select two of them. If you want to volunteer, please raise your hand and I will write down your name.
Thank you.
Appendix E

Student Interview Protocol

1. This is the computer model that you built. Please describe this model and tell me how it explains the physical phenomena (free fall, motion in one direction, action and reaction force pairs, etc.)

2. What would you change to improve this model?

3. What did you find the most challenging when you were building this model?

4. Do you think engaging in modeling activities helped you understand the Newton's Laws of Motion better?

5. If so, in what ways it helped you understand Newton's Laws of Motion better? What did you learn or understand better during the activities?

6. Did you enjoy learning about computational thinking and computer programming? What aspect(s) of the learning activities did you enjoy in particular and why?

7. Did you enjoy building computer models? What aspect(s) of the modeling activities did you enjoy in particular and why?

8. Would you like to do this kind of modeling activities more in your science classrooms?

9. Is there anything else you would like to share with me today?