Journal of Science Education and Technology (2020) 29:65–82
https://doi.org/10.1007/s10956-019-09800-z

Exploring Force and Motion Concepts in Middle Grades Using Computational Modeling: a Classroom Intervention Study

Osman Aksit[1] · Eric N. Wiebe[2]

Published online: 28 November 2019

Abstract

Computational thinking (CT) and modeling are authentic practices that scientists and engineers use frequently in their daily work. Advances in computing technologies have further emphasized the centrality of modeling in science by making computationally enabled model use and construction more accessible to scientists. As such, it is important for all students to get exposed to these practices in K-12 science classrooms. This study investigated how a week-long intervention in a regular middle school science classroom that introduced CT and simulation-based model building through block-based programming influenced students' learning of CT and force and motion concepts. Eighty-two seventh-grade students from a public middle school participated in the study. Quantitative data sources included pre- and post-assessments of students' understanding of force and motion concepts and CT abilities. Qualitative data sources included classroom observation notes, student interviews, and students' reflection statements. During the intervention, students were introduced to CT using block-based programming and engaged in constructing simulation-based computational models of physical phenomena. The findings of the study indicated that engaging in building computational models 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. This study contributes to the nascent literature on integrating CT into K-12 science curricula by emphasizing the affordances and generative dimension of model construction through block-based programming.

Keywords Computational thinking · Modeling · Block-based programming · Force and motion · Middle school · Science

Introduction

Computational thinking (CT) has become an increasingly important skill and core competency in all science, technology, engineering, and mathematics (STEM) fields (National Research Council [NRC] 2010, 2011). As such, the Next Generation Science Standards (NGSS) identified CT as one of the eight science and engineering key practices that are essential for all K-12 students to engage in as part of their science learning in the classroom (NGSS Lead States 2013). Although there is debate in the scholarly community on the definition of CT and what specific concepts, abilities, and practices CT involves (Grover and Pea 2013), researchers and policymakers have argued that CT has become an indispensable competency that all students must acquire as part of their K-12 education (Barr and Stephenson 2011; K-12 Computer Science [CS] Framework Steering Committee 2016; NRC 2012).

Despite the apparent consensus on CT as a key set of concepts and practices central to all STEM disciplines, research suggests that CT has not been integrated well into K-12 science curricula nor addressed properly in science classrooms (Sengupta et al. 2013; Voogt et al. 2015; Wilensky et al. 2014). Existing research efforts have mostly focused on developing CT competencies in open-ended contexts including game design and storytelling with programming, or in informal settings through extracurricular and after-school activities (e.g., Bers 2010; Bers et al. 2014; Denner et al. 2012). New lines of research have demonstrated that model-based pedagogy can be used as an effective medium to introduce and facilitate CT

✉ Osman Aksit
osmanaksit@gmail.com

Eric N. Wiebe
wiebe@ncsu.edu

[1] Dhahran Ahliyya Schools, PO Box 39333, Dhahran 31942, Saudi Arabia

[2] Department of STEM Education, North Carolina State University, Box 7801, 326 Poe Hall, Raleigh, NC 27695, USA

development in addition to fostering science learning in formal classroom settings (Basu et al. 2016; Hambrusch et al. 2009; Sengupta et al. 2013; Weintrop et al. 2016; Wilensky et al. 2014).

The use of programming languages as a modeling tool for the creation of simulations for the exploration of science concepts has been demonstrated as a powerful vehicle for learning, especially in the physical sciences (diSessa 2000; Sherin et al. 1993; Wilensky and Resnick 1999). However, learning the syntax and semantics of a text-based programming language is commonly one of the most frustrating challenges for new learners (Guzdial 2004), especially when it is happening simultaneously with grappling with new science concepts. Block-based programming is a relatively new paradigm compared to the traditional text-based programming and was specifically designed to introduce programming to beginners. The 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 CT (Lye and Koh 2014).

Thus, the emergence of block-based programming has led to a renewed interest in the still understudied set of ideas around using CT-focused scientific modeling activities as a vehicle to directly facilitate and foster both science learning and CT development in formal classroom settings (Grover and Pea 2013; Basu et al. 2016). This is especially true for younger aged children since the older computational modeling environments required the learning of more complex text-based programming languages. Parallel with the renewed interest in computational modeling in science has been the development of instruments to measure the relatively new construct of CT (e.g., Román-González et al. 2016). These instruments provide a relatively direct means for researchers to explore the efficacy of block-based computational modeling to develop CT.

In this study, we designed and conducted a week-long classroom intervention in a public middle school where students engaged in learning about block-based programming and constructed computational models of physical phenomena in the context of force and motion. We report our findings regarding students' gains on CT and force and motion concepts. The lessons learned from the design of the learning activities and the actual classroom implementation are also discussed to inform future research.

Literature Review

CT in K-12 Education

While there has been an ongoing discussion in the research community regarding the definition of CT (Grover and Pea 2013) and its relationship with other types of analytical competencies such as mathematical thinking (Sneider et al. 2014), this study conceptualized CT as the "thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (Aho 2012, p. 832). CT 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 Steering Committee, 2016). Lu and Fletcher (2009) argued that CT 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).

An important line of work to develop a CT assessment independent both of specific programming languages and curricular contexts has been led by Román-González et al. (2016, 2018). They chose to take a psychometric approach rooted in the CHC model of intelligence (McGrew, 2009). Román-González et al. (2016) has conjectured a relationship between CT knowledge and ability constructs such as abstraction, pattern generalization, algorithmic thinking, and conditional logic, and the CHC constructs of fluid intelligence, visual processing, and working memory. The design of these items was informed both by prior assessments developed around CS-centric programming tasks as well as other efforts developing programming-independent assessments (Lockwood and Mooney 2018). This assessment, the Computational Thinking Test (CTt), provides a decontextualized assessment of CT based not on specific programming knowledge, but of constructs grounded in the CHC model of cognitive abilities. An open question with this new assessment is its level of diagnostic sensitivity to short-term interventions based around computational modeling.

To help guide the development of curricula and curricula-based assessments of CT, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) collaborated to form a steering committee and identify CT concepts and practices in the context of K-12 education (Barr and Stephenson 2011). The committee identified nine essential concepts and practices pertaining to CT, which include "data collection, data analysis, data representation, problem decomposition, abstraction, algorithms, automation, parallelization, and simulation" (Barr and Stephenson 2011). The committee also provided examples of pedagogical strategies for embedding CT 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 collected from an experiment in a science classroom and in discovering patterns for different sentence types in a language art classroom (Barr and Stephenson 2011).

The development of CT curricular frameworks has helped guide the beginning of new line of research into the efficacy of computer programming to develop CT abilities in formal K-12 settings (Voogt et al. 2015). Likewise, these frameworks have also opened a new line of research into optimal instructional strategies for helping K-12 students engage in CT as part of their science learning in formal classroom settings (Sengupta et al. 2013). Combined with new tools for measuring CT development at a more general, cognitive level, there is now the opportunity to look at how CT-focused computational modeling activities in science classrooms might forward both science learning and CT abilities.

Learning About Force and Motion Concepts

The concept of force is one of the core concepts in physics, which is used to describe the behavior of objects in electromagnetic and gravitational fields. A conceptual understanding of force is important for describing how interaction between objects leads to a change in their motion (NRC 2012). Students' conceptual understanding of force and motion concepts has been extensively studied in the literature (e.g., Alonzo and Steedle 2009; Eryilmaz 2002; Hestenes et al. 1992; Kozhevnikov and Thornton 2006; Reiner et al. 2000; Yuruk et al. 2009). Previous research has 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 et al. 2013). For example, students tend to think that there should be a force acting on an object if the object is moving (Halloun and 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 and Hestenes 1985). Some students think that velocity of an object is directly proportional to the applied force; therefore, constant force means constant velocity (Sequeira and 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 (Alonzo and Steedle 2009; Yuruk et al. 2009). 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 (2002) 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).

The deep literature base in physics conceptual understanding provides guidance as to which concepts middle grade students are likely to struggle with and insight as to what interactive, computer-based affordances might help support their learning. The next section will explore literature on computer-based modeling and how it has been used to support learning in this science area.

Modeling and CT

Modeling plays a central role in the conduct of scientific research (Gilbert 1991; Schwarz and White 2005). Most scientists spend a great deal of their time in building, testing, studying, and revising models, often using computational tools. 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 and Treagust 1998; Leenaars et al. 2013). A scientific model can be a conceptual, mathematical, or physical representation of physical phenomena. Models are built by using a variety of tools such as plots, diagrams, physical structures, computer programs, and simulations (Bailer-Jones 2003). 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 and Buckley 2000; Delgado and 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 and van Joolingen 1998; diSessa 2000). As students must develop skills for constructing, interpreting, and coordinating domain-specific representations for learning and problem solving, they develop a broad, powerful set of learning tools referred to as representational competence (diSessa 2004; Wu and Puntambekar 2012). In computational modeling activities, where students grapple with representing scientific concepts in programming code with the goal of creating visual output representing the same concepts, there is considerable potential for developing such meta-

representational competencies (diSessa 2004; Sengupta et al. 2013).

The K-12 CS Framework (K-12 CS Framework Steering Committee 2016), which was developed jointly by the Computer Science Teachers Association (CSTA) and the Association for Computing Machinery (ACM), recommends integrating CT 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' CT practices can be synergistically supported by modeling in the science classroom (Basu et al. 2016; Sengupta et al. 2013).

In a study targeting middle and high school students, Sengupta et al. (2012) designed a series of simulation-based scientific models with a visual programming environment called ViMAP and investigated how engaging in these activities supported students' learning of kinematics concepts. Based on the NetLogo programming environment, it could be considered as leveraging some of the same visual-based scaffolding of current block-based programming environments such as Scratch (Resnick et al. 2009). The researchers were interested in "integrating programming and computational modeling seamlessly with classroom physics instruction in the particular domain of kinematics" (Sengupta et al. 2012, p. 24). 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).

Using similar tools, Basu et al. (2012) worked with a sample of sixth-grade students (N = 24) to investigate whether engaging in modeling through a customized visual programming environment helped students improve their understanding of kinematics and ecology concepts. While this study did show significant science learning gains for the partially scaffolding group, the researchers did not investigate how the intervention influenced students' CT ability. Finally, Broll et al. (2017) demonstrated a new computational modeling environment based on Snap!, a contemporary block-based programming tool. Though they demonstrated the powerful, cognitively supportive visual modeling environment well suited to explore physics phenomena, no student learning outcomes were presented.

In summary, there has been important work emerging building off of foundational computational modeling work, re-envisioned for use with contemporary visual, block-based programming environments. To date, most of the work has been demonstration projects or focused on science learning outcomes (e.g., Basu et al. 2012). Similarly, block-based programming tools and new CT curricular learning frameworks have been used to explore development of CT abilities (e.g., Grover et al. 2016), but little has been done to look at both science learning and CT development in the same block-based computational modeling environment.

Theoretical Framework

The Representational Construction Affordances (RCA) framework (Prain and Tytler 2012) served as the theoretical framework of this study. The RCA framework recognizes 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." (Prain and Tytler 2012, p. 2756). The authors built off of the earlier work of diSessa (2004) and others who demonstrated the importance of developing meta-representational competencies but emphasized not only the manipulation of representations, but also the creation of them, in particular visual representations. Meta-representational competencies speak to the need to provide students opportunities to develop their ability to critique, select, and create different representations that forward their problem solutions (diSessa 2004). For example, in traditional classroom instruction, students may be asked to use a simulation representing a predator-prey model without learning how to develop the model and construct the simulation, as well as reflecting on the limitations posed by the simulation in representing the scientific ideas.

In the RCA 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). The second two dimensions are perhaps most relevant for this work. The epistemic dimension emphasizes the knowledge-building practices such as constructing models, and making, justifying, and communicating claims based on the constructed models. 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 external representations in science classrooms leads to high learning gains because building representations is not only very engaging but also improves students' reasoning strategies and representational competence as well as helps students better communicate and articulate their understanding with their peers and teachers.

In introducing block-based programming as an additional representational form, this study's modeling practice provides an opportunity for students to work with another

representational form (i.e., programming blocks) of the scientific concepts (Wilkerson-Jerde et al. 2015). This additional representational form may provide unique affordances for conceptual interpretation for some students. Perhaps equally important is providing the opportunity for students to develop meta-representational competence of constructing ideas in multiple forms (diSessa 2004) and allowing students to compare and contrast the strengths of each representational form—meta-modeling ability (Schwarz et al. 2009).

Purpose of the Study

The goal of this study was to investigate how participating in a week-long intervention course that introduced CT practices and simulation-based model building through a block-based programming environment in regular middle school science classrooms influenced seventh-grade students' CT abilities as well as their conceptual understanding of force and motion concepts. Since there is a dearth of research on how to integrate CT into science curricula and instruction in formal K-12 settings, the findings of this study provide insights on how students' CT abilities and science learning develop by engaging in block-based programming and model-based pedagogy. Specifically, this study aimed to answer the following research questions:

1. How does building computational models of physical phenomena through a block-based programming environment in a week-long intervention course influence middle school students' conceptual understanding of force and motion concepts?
2. How does participating in a week-long intervention course introducing CT and computational modeling through a block-based programming environment influence middle school students' CT abilities?

Methodology

Setting

This study took place at a public middle school in the southeastern USA 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 courses on basic computer science topics. The school was selected for this study because of the authors' familiarity with the school and school district. The classroom intervention was conducted by the first author in one of the computer labs in the school during the regular seventh-grade science class periods (50 min each).

Participants

A seventh-grade science teacher who was teaching five different sections was recruited to participate in this study. The total number of students in the teacher's classrooms was 132 and all students were invited to participate in the study. Eighty-two students consented to participate. While all students attended the classroom intervention and engaged in the learning activities, only the data from the consenting students was used in this study. Participation to the study was completely voluntary and had no effect on students' school grade. Fifty-four percent of the participating students were male ($n = 44$). 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.

Research Design

This study employed a convergent mixed methods research design, using a combination of quantitative and qualitative research approaches (Creswell 2015). Figure 1 shows the design of this study, illustrating the flow of activities and procedures at each stage.

Intervention

The purpose of the classroom intervention designed for this study was to introduce seventh-grade students to CT concepts and practices in the context of building simulation-based models of physical phenomena through a block-based programming environment in formal classroom settings. The intervention aimed to further students' CT abilities as well as conceptual understanding of force and motion concepts. The intervention was carefully designed by the first author using their past experience as a physics teacher as well as their background in teaching introductory-level programming. Scratch (https://scratch.mit.edu) was chosen as the block-based programming environment that students worked with during the classroom intervention. Scratch was developed at the Massachusetts Institute of Technology as a freely available open-source software to teach computer programming through a visual, block-based environment (Maloney et al. 2010).

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 CT practices (e.g., abstraction, algorithms) as well as the fundamental

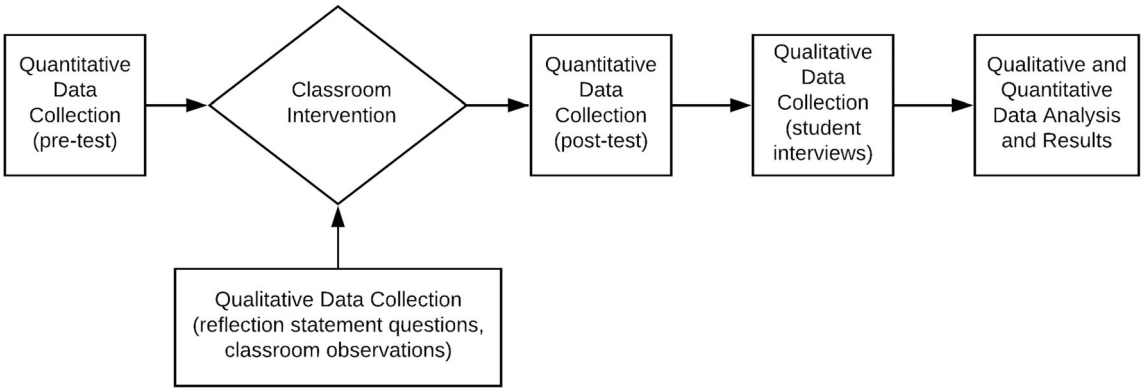

Fig. 1 Research design of the study

concepts in computer programming (e.g., variables, conditionals, loops) using the Scratch environment. In the next two days, students engaged in simulation-based model building activities in Scratch to demonstrate and improve their conceptual understanding of force and motion concepts. It is important to note here that students formally studied force and motion unit in the previous semester, so the purpose of the modeling activities was to address any misconceptions students might have had and further improve their conceptual understanding of force and motion concepts. Table 1 shows the contents of the learning activities covered for each day.

Figure 2 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.

Figure 3 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., a basketball) and the stage (i.e., a background picture with a ground represented by a black line at the bottom), and they were asked to develop an algorithm and construct a script for this model, just like they did in the previous day.

Data Sources

Data for this study were collected from a variety of sources including two multiple-choice assessments, students' written responses to reflective statement questions, detailed classroom observation notes, and interviews with students.

The pre- and post-test included a 28-item multiple-choice assessment called the Computational Thinking Test (CTt; Román-González et al. 2016) that aimed to assess students' CT abilities and a 16-item multiple-choice assessment called the Force and Motion Assessment (FMA; Alonzo and Steedle 2009) that aimed to assess students' conceptual understanding of force and motion concepts.

During the classroom intervention, students were asked to reflect on what they learned by responding to the reflective statement question of "What did I learn today?" at the end of each class period before leaving the classroom.

In addition to the first author who conducted the intervention, a member from the research team was always present in the classroom and took extensive classroom observation notes. These notes helped assess the fidelity of the intervention, i.e., described and documented how the learning activities went in the classroom, whether

Table 1 Computational thinking concepts and modeling activities covered during the intervention

Day 1	Introduction to block-based programming and Scratch user interface; problem decomposition and creating algorithms as a set of instructions; creating and using abstractions; conditionals and decision-making; using mathematical operators.
Day 2	Loops and repetition structures; nested loops and conditionals; using logical operators.
Day 3	Creating, using, and manipulating different types of variables; event-driven programming with broadcasting in Scratch.
Day 4	Model 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.
Day 5	Model building activity 2—building a model that will simulate the fall of a basketball in a 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.

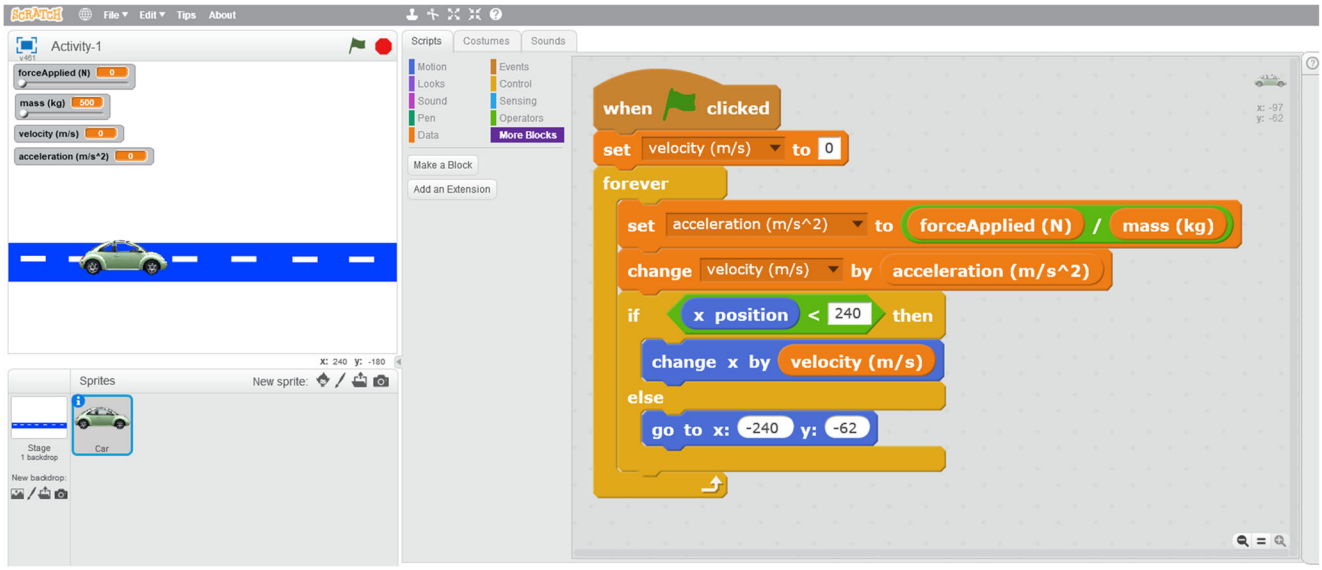

Fig. 2 Screenshot of the first modeling activity

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 authors understand students' engagement in the learning activities.

After the completion of the classroom intervention, the first author conducted structured interviews with a subset of consenting students (n = 12). Participation to the interview was voluntary and students who volunteered received a $10 gift card. 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 to further explore their conceptual understanding of force and motion concepts as well as their understanding of CT concepts and practices introduced during the classroom intervention. The researcher audio-recorded the interviews and transcribed them verbatim for analysis.

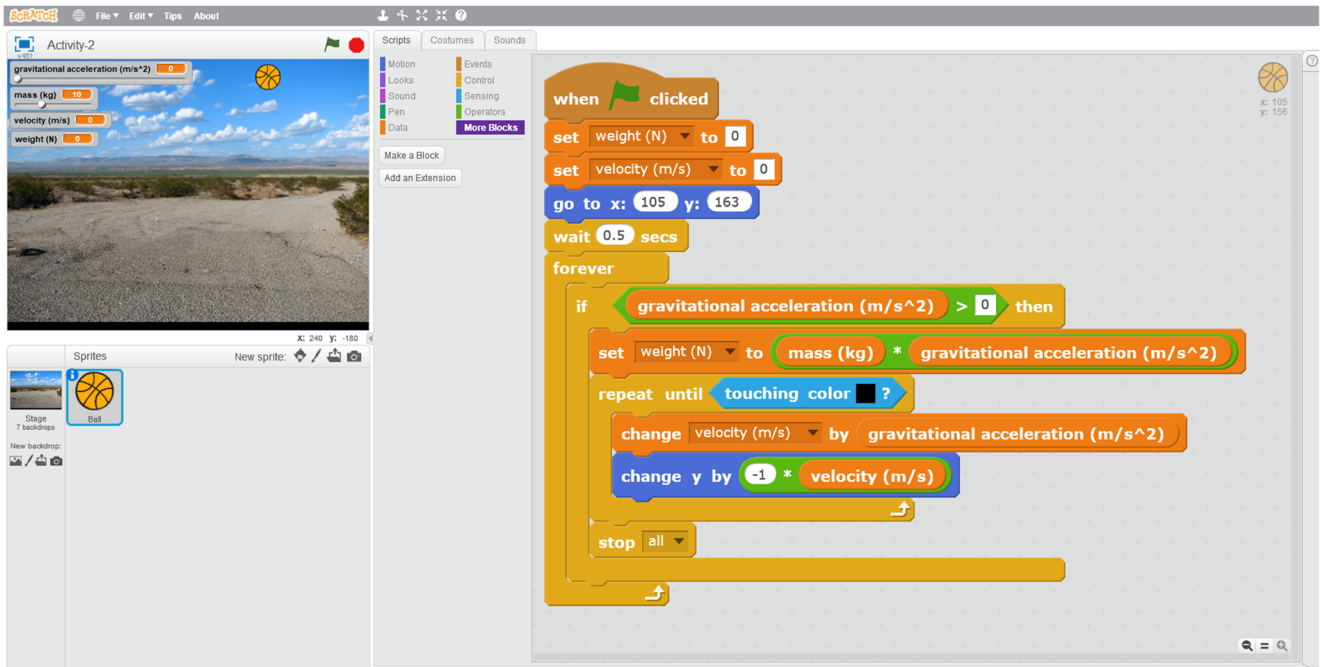

Fig. 3 Screenshot of the second modeling activity

Data Analysis

The collected data was analyzed using both quantitative and qualitative methods. Winsteps 4.0 (Linacre 2018) software was used to conduct Rasch analysis (Rasch 1980) and generate log odd units (logit) for students' scores for the multiple-choice assessments on the pre- and post-test. Rasch analysis takes into account the difficulty of the assessment items when generating student scores in terms of logits, which provides a more accurate measurement of the latent trait (e.g., conceptual understanding) (Boone et al. 2011). IBM SPSS Statistics 23 was used to run the descriptive statistics and the statistical tests. Only the student scores on the Rasch scale were 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). Descriptive statistics for students' raw scores as well as their scores on the Rasch scale (logits) for the multiple-choice assessments were generated for both the pre- and post-test.

Qualitative data analysis was carried out by two researchers, the first author and a doctoral student. First, both researchers carefully read the students' responses and developed their own categories individually using an open coding approach (Creswell and Clark 2011). 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 re-coded 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, $\kappa = 0.786$ (95% CI, 0.752 to 0.820, $p < 0.001$).

Results and Discussion

Research question 1

How does building computational models of physical phenomena through a block-based programming environment in a week-long intervention course influence middle school students' conceptual understanding of force and motion concepts?

Students' pre- and post-test scores for the FMA (Alonzo and 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 2 shows descriptive statistics of the students' raw scores for the FMA 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 FMA 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 (out of 100) 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 2). Figure 4 shows histograms of frequencies of students' scores on Rasch scale for the FMA on the pre- and post-test.

A paired samples t test was conducted using students' pre- and post-test scores on the Rasch scale for the FMA 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 3, showed that participating in the classroom intervention 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, $p < 0.001$, $d = 1.02$).

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 they 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 concept 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 Jr and Sweet 2009; Hestenes et al. 1992). Secondly, this student's response also suggests that they 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 and 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

Table 2 Descriptive statistics for students' scores on the Force and Motion Assessment

	M	*SD*	Median	Mode	Min.	Max.	Range
Force and Motion Assess.—pre[a]	4.01	2.17	3.5	3	0	9	9
Force and Motion Assess.—post[a]	6.82	2.96	6	5	2	13	11
Force and Motion Assess.—pre (logit)[b]	33.99	10.21	33.33	30.98	0.17	53.22	53.05
Force and Motion Assess.—post (logit)[b]	45.98	10.63	43.60	40.00	25.17	68.98	43.81

$N = 82$

[a] Raw scores out of 16

[b] Scaled to 1–100

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 they engaged in the modeling activities. Like most of the other students in the classroom, they 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 their 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 and McDermott 1987; Dykstra Jr and Sweet 2009; Hestenes 2007). In an earlier study with middle and high school students, Jones (1983) found that the following alternative conceptions were prevalent among students: increasing speed always means increasing acceleration, and if an object is going fast then it has acceleration. Students often confuse acceleration with velocity and speed, thinking that if two objects have the same speed, then they should have the same acceleration (Rosenquist and McDermott 1987; Dykstra and Sweet 2009; Tasar 2010). Perhaps not

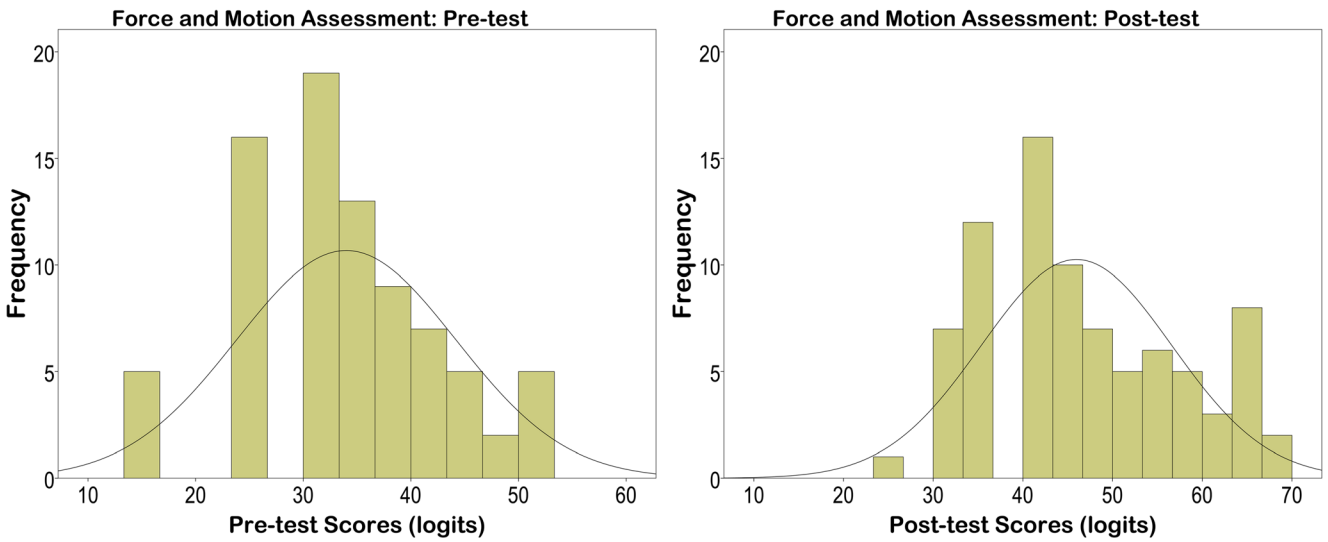

Fig. 4 Histograms of frequencies of students' FMA scores on Rasch scale

Table 3 Results of the paired samples *t* test analysis comparing students' pre- and post-test scores on the FMA

	Pre-test		Post-test			
	M	*SD*	*M*	*SD*	*t* (81)	Cohen's *d*
FMA (logit)[a]	33.99	10.21	45.98	10.63	9.26***	1.02

$N = 82$

***$p < 0.001$, two-tailed

[a] Scaled to 1–100

surprisingly, similar to the findings of the previous studies, participating students of this study struggled explaining and conceptualizing acceleration during the classroom intervention. Since the force and motion unit was covered in the previous semester, some students were able to recite a definition that they had probably memorized for their exam. However, when the first author 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 many 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 students' 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, they were able to understand that acceleration made the car "gradually became faster and faster." Here is another student from the interviews that defined acceleration in their own words, which illustrates their 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 model building activities 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 (i.e., two days) in the classroom on this topic? We argue that the answer to this question is "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 FMA on the pre-test was $M = 33.99$ out of 100 on the 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 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 through block-based programming 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 force and motion concepts, used their models to articulate and convey their understanding of the concepts to their peers, and discussed if their models 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?," and "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 also enabled students to explore force and motion concepts through creating and working with multiple representational systems (diSessa 2004; Wilkerson-Jerde et al. 2015) 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. Each representation has its own specific affordances and constraints. For example, Fig. 5 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, students 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, students 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 et al. 1993; Sherin 2001; Wagh et al. 2017). For example, Fig. 6 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.

Lastly, block-based programming 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 × acceleration" 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 et al. 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., Fig. 6). 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 Fig. 6, 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.

In accordance with earlier literature on the efficacy of interactive modeling environments (e.g., diSessa 2000, 2004; Tisue and Wilensky 2004), the Scratch block-based computational modeling environment allowed students to engage deeply with problems of understanding and representing

Fig. 5 Mathematical formula versus block-based code on Scratch

Fig. 6 Calculating velocity using block-based code

conceptually important physics problems. That is, constructing computational models using block-based programming requires students to think and formulate the rules (i.e., science concepts) that govern the behavior of a physical phenomenon. Next, the dynamic visual output of the simulation allows students to test and observe how their code (i.e., algorithm) behaves in real time. Finally, block-based programming potentially has an advantage over earlier computational modeling and representational systems by virtue of the research and development of block-based visual programming environments that have minimized the syntactic overhead of text-based programming tools (cf., Weintrop and Wilensky 2015, 2017).

Research question 2

How does participating in a week-long intervention course introducing CT and computational modeling through a block-based programming environment influence middle school students' CT abilities?

Students' pre- and post-test scores for the CTt were used to investigate their CT abilities before and after the intervention. Internal consistency of the CTt was measured using Cronbach's alpha and was found to be $\alpha = 0.772$. Table 4 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 (out of 100) on the Rasch scale. Figure 7 shows histograms of frequencies of students' scores on the Rasch scale for the CTt on the pre- and post-test.

A paired samples t test was conducted using students' pre- and post-test scores on the Rasch scale to determine if students' CT abilities significantly differed before and after the classroom intervention. The results of the paired samples t test, as presented in Table 5, showed that participating in the intervention course resulted in a statistically significant increase in seventh-grade students' CT abilities with a large effect size ($t(81) = 8.06$, $p < 0.001$, $d = 0.91$).

Overall, the results of the pre-test indicate that prior to the classroom intervention, students' CT abilities were moderate ($M = 58.37$, $SD = 12.77$; out of 100 on the Rasch scale). The histogram of frequencies of students' pre-test scores on the Rasch scale shows that majority of the students' responses were clustered between 40 and 80 with 63 as the mode of the distribution (Fig. 7), which may be related to prior exposure to computational activities. In fact, many students stated during the 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 help explain participating students' overall modest performance on the pre-test.

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 CT abilities in the post-test, with their average post-test score on the Rasch scale $M = 65.33$ ($SD = 11.05$). 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 (Fig. 7). The significant gains on CTt during the intervention indicate that students can achieve higher CT ability levels if computationally rich activities are 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 the Scratch programming environment. Due to time constraints, it was not possible to do this during the intervention. However, the authors recommend 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 CT concepts and develop CT abilities in general. It is important to note that the first author 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 CTt scores meant that the growth in the area of CT ability was modest. Future studies conducting longer term interventions can assess students'

Table 4 Descriptive statistics for students' scores on the CTt

	M	*SD*	Median	Mode	Min.	Max.	Range
CTt—pre[a]	17.83	5.63	19	21	6	28	22
CTt—post[a]	20.76	4.29	22	22	11	28	17
CTt—pre (logit)[b]	58.37	12.77	58.98	63.11	34.12	99.93	65.81
CTt—post (logit)[b]	65.33	11.05	66.22	66.22	45.06	99.97	54.91

$N = 82$

[a] Raw scores out of 28

[b] Scaled to 1–100

prior CT ability and then adjust the programming activities to differentiate the instruction and allocate more time across learning activities related to science.

To complement 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 learning and application of CT concepts and practices. Here is an example of a student's response to the reflective statement question of "What did I learn today?" showing their understanding of conditionals and decision-making in computer programming: "I learned that to take out many options or commands, then you would use the if then, else block." By saying "many options," the student was probably referring to the different conditions they coded in their programs to make the sprite take different decisions 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 Steering Committee 2016). Here is a reflective statement response of another student that indicated their 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 and 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 et al. 2003; Grover et al. 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 their 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 their 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 their understanding of the *repeat* and *repeat until* looping structures: "I learned how to

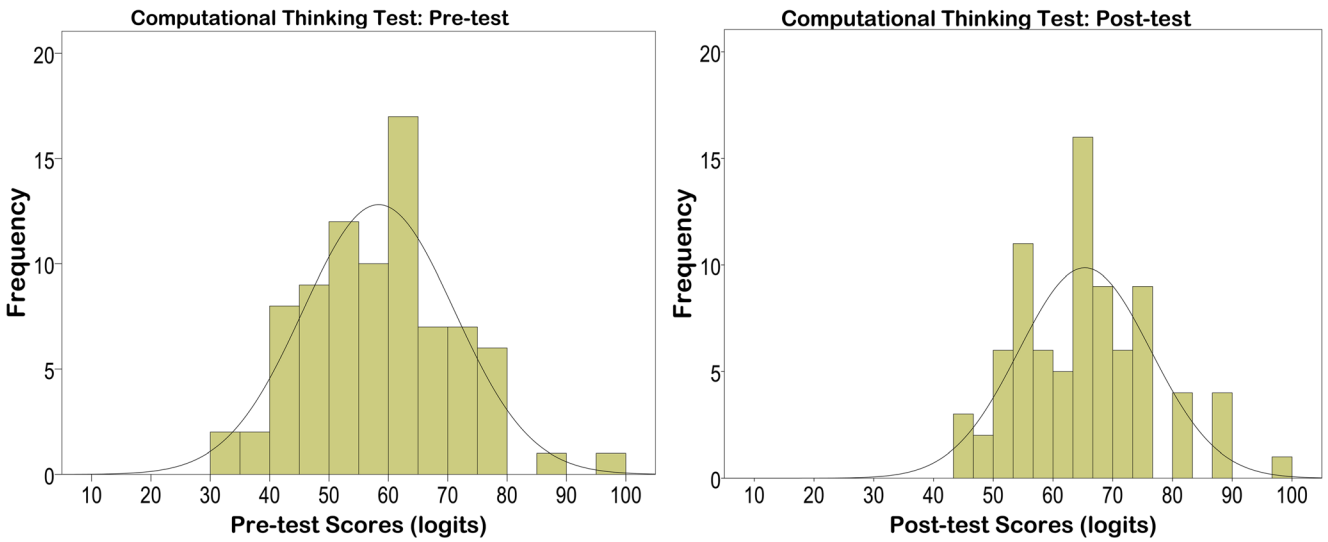

Fig. 7 Histograms of frequencies of students' CTt scores on Rasch scale

Table 5 Results of the paired samples *t* test comparing students' pre- and post-test scores on the CTt

	Pre-test		Post-test			
	M	*SD*	*M*	*SD*	*t* (81)	Cohen's *d*
CTt (logit)[a]	58.37	12.77	65.33	11.05	8.06***	0.91

$N = 82$

***$p < 0.001$, two-tailed

[a] Scaled to 1–100

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 authors identified several challenges faced by the participating students during the classroom intervention using the classroom observation notes and students' interview responses. Firstly, most of the participating students struggled outlining and formulating an algorithm when they were given a programming task such as building a pong game. Devising an algorithm requires examining the target problem and breaking it down into 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 Steering Committee 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 first author 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 then devising 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 on their computers. However, the observation notes indicated 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 be 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. Researchers recommended engaging students in creating pseudocodes to draft an algorithmic solution to the given problem during introductory programming exercises (Futschek 2006; Grover et al. 2014).

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 (commonly referred to 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. 47). Debugging is a crucial skill in programming and requires complex reasoning (Grover et al. 2016). The classroom intervention of this study did not explicitly teach students about debugging. However, the researcher briefly 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 also 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 advanced 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, and the instructor needs to make an informed decision based on the available time allocated for these activities. If the CT activities 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.

Limitations

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 USA. The

sample of this study may not be representative of the seventh-grade student population in the USA. This may limit the generalizability of the findings of this study to a larger student population. Also, 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 gains were due to the classroom intervention alone.

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 particularly interested in block-based programming and actively engaged in the learning activities during the classroom intervention. Therefore, their responses may not necessarily be representative of the larger group of students who participated in this research. 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. 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 language was English.

The intervention of this study lasted only for five class periods and employed an immediate post-test to assess student learning during the classroom intervention. Thus, our study 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' learning of science and CT concepts. Similarly, it would be useful to provide students similar computational modeling opportunities on a regular basis to explore the potential sustained positive affective and cognitive shift over extended periods of time (e.g., multiple school years).

The CTt, which was used to assess participating students' CT abilities prior to and after the classroom intervention, entirely consisted of multiple-choice questions. The developers of the test, Román-González et al. (2016), pointed out that the CTt "might be measuring CT at its lower cognitive complexity levels ('recognize' and 'understand')" (p. 688). Therefore, this study might not have assessed students' CT ability at higher levels of cognitive complexity. Lastly, the learning activities of this study were not differentiated based on students' prior experiences in block-based programming. 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.

Conclusion and Future Work

CT and modeling are authentic practices that scientists and engineers use frequently in their daily work (NRC 2012). Therefore, it is crucial for students to get exposed to these practices in their science classrooms. The findings of this study indicate that infusing CT concepts and practices into science instruction in the context of computational modeling activities can foster students' understanding of both CT and science concepts and also develop general CT abilities, as demonstrated through the pre-post assessments and student interviews. The findings of this study also suggest that even with minimal time in the classroom to introduce CT concepts and practices through block-based programming, this can still enable students to build relatively advanced computational models, which, in turn, can facilitate conceptual science learning in formal classroom settings. This study contributes to the evolving literature on integrating CT into science instruction by emphasizing the affordances and generative dimension of model construction through block-based programming.

With regard to the impact of computational modeling on science learning, we were not able to disentangle the affordances of interactive simulations from the generative representational activity of constructing computational models. This would be important since already constructed simulations have their own unique learning affordances (cf., Rutten et al. 2012). Similarly, if Román-González et al.'s (2016) conjecture that spatial ability is an important component of CT, then we would also need to be concerned with what component simply interacting with the spatial elements of the computational model might have developed CT-related spatial ability. Finally, we did not extensively explore how infusing CT into science instruction through a model-based pedagogy affects classroom discourse. The classroom observation notes indicated that there was a rich discussion among students regarding 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, in conjunction with model construction and manipulation, helps students develop both science concepts and CT abilities.

Since classrooms are getting more diverse nationwide with increasing number of ELL students, future studies should explore what kind of scaffolds can be provided by teachers as well as by the programming environment to ELL students during programming and modeling activities, which can help them overcome language barriers and fully participate in learning activities. This can be particularly helpful for students whose native tongue is English but have reading or learning

difficulties. Leveraging the spatial nature of computational models and simulations with culturally relevant representations and minimizing dense, specialized text in curriculum design would provide more accessible learning opportunities for ELL students.

The intervention of this study was conducted by the first author. Future research should investigate prospective and in-service teachers' self-efficacy towards using computational tools and model-based pedagogy in the classroom to better understand teacher-curricular interactions. If this approach of introducing CT in science classrooms in the context of computational modeling 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' competencies and self-efficacy with regard to programming and CT practices and its interaction with classroom practices when they are implementing a computational modeling-based pedagogy.

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Ethical Approval All procedures performed in this study were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors.

Informed Consent Informed consent was obtained from all individual participants included in the study.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835.

Alonzo, A. C., & Steedle, J. T. (2009). Developing and assessing a force and motion learning progression. *Science Education, 93*(3), 389–421.

Bailer-Jones, D. M. (2003). When scientific models represent. *International Studies in the Philosophy of Science, 17*(1), 59–74.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning, 11*(13), 1–35.

Basu, S., Kinnebrew, J., Dickes, A., Farris, A. V., Sengupta, P., Winger, J., & Biswas, G. (2012). A science learning environment using a computational thinking approach. In *Proceedings of the 20th International Conference on Computers in Education* (pp. 722-729).

Bers, M. U. (2010). The TangibleK Robotics program: applied computational thinking for young children. *Early Childhood Research and Practice, 12*(2). Retrieved November 12, 2019, from https://files.eric.ed.gov/fulltext/EJ910910.pdf

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

Boone, W. J., Townsend, J. S., & Staver, J. (2011). Using Rasch theory to guide the practice of survey development and survey data analysis in science education and to inform science reform efforts: an exemplar utilizing STEBI self-efficacy data. *Science Education, 95*(2), 258–280.

Boulter, C. J., & Buckley, B. C. (2000). Constructing a typology of models for science education. In *Developing models in science education* (pp. 41-57). Springer Netherlands.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, Canada.

Broll, B., Zare, H., & Ledeczi, A. (2017). *Creating engaging science projects with netsblox.* Paper presented at the 2017 IEEE Blocks and Beyond Workshop (B&B).

Casey, P. J. (1997). Computer programming: a medium for teaching problem solving. *Computers in the Schools, 13*(1-2), 41–51.

Creswell, J. W. (2015). *A concise introduction to mixed methods research*. Sage Publications.

Creswell, J. W., & Clark, V. L. P. (2011). *Designing and conducting mixed methods research*. Sage Publications.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

de Jong, T., & van Joolingen, W. R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research, 68*(2), 179–201.

Delgado, C., & Krajcik, J. (2010). Technology and learning—supports for subject matter learning. In *International encyclopedia of education* (3rd ed., pp. 197–203). Oxford: Elsevier.

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

diSessa, A. A. (2000). *Changing minds: computers, learning, and literacy*. Boston: MIT Press.

diSessa, A. A. (2004). Metarepresentation: native competence and targets for instruction. *Cognition and Instruction, 22*(3), 293–331.

diSessa, A. A., & Sherin, B. L. (1998). What changes in conceptual change? *International Journal of Science Education, 20*(10), 1155–1191.

Dykstra Jr., D. I., & Sweet, D. R. (2009). Conceptual development about motion and force in elementary and middle school students. *American Journal of Physics, 77*(5), 468–476.

Eryilmaz, A. (2002). Effects of conceptual assignments and conceptual change discussions on students' misconceptions and achievement regarding force and motion. *Journal of Research in Science Teaching, 39*(10), 1001–1015.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In *In International Conference on Informatics in Secondary Schools - Evolution and Perspectives* (pp. 159–168). Berlin, Heidelberg: Springer.

Gilbert, S. W. (1991). Model building and a definition of science. *Journal of Research in Science Teaching, 28*(1), 73–79.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 57-62). ACM.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Grover, S., Pea, R., & Cooper, S. (2016). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM*

Technical Symposium on Computing Science Education (pp. 552-557). ACM.

Guzdial, M. (2004). Programming environments for novices. In S. A. Fincher & M. Petre (Eds.), *Computer science education research* (pp. 127–154). London: Taylor & Francis Group, PLC.

Halloun, I. A., & Hestenes, D. (1985). Common sense concepts about motion. *American Journal of Physics, 53*(11), 1056–1065.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)* (pp. 183–187). New York: ACM.

Harrison, A. G., & Treagust, D. F. (1998). Modelling in science lessons: are there better ways to learn with models? *School Science and Mathematics, 98*(8), 420–429.

Hestenes, D. (2007). Modeling theory for math and science education. In *Paper presented at the ICTMA13: The International Community of Teachers of Mathematical Modelling and Applications*. Indiana: IL.

Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher, 30*(3), 141–158.

Ioannides, C., & Vosniadou, S. (2002). The changing meanings of force. *Cognitive Science Quarterly, 2*(1), 5–62.

Jones, A. T. (1983). Investigation of students' understanding of speed, velocity and acceleration. *Research in Science Education, 13*(1), 95–104.

K-12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. Retrieved November 12, 2019, from https://www.k12cs.org

Kozhevnikov, M., & Thornton, R. (2006). Real-time data display, spatial visualization ability, and learning force and motion concepts. *Journal of Science Education and Technology, 15*(1), 111–132.

Leenaars, F. A., van Joolingen, W. R., & Bollen, L. (2013). Using self-made drawings to support modelling in science education. *British Journal of Educational Technology, 44*(1), 82–94.

Linacre, J. M. (2018). *Winsteps Rasch measurement computer program*. Beaverton: Winsteps.com.

Lockwood, J., & Mooney, A. (2018). *Developing a Computational Thinking Test using Bebras problems.* Paper presented at the Joint Proceedings of the CC-TEL 2018 and TACKLE 2018 Workshops, co-located with 13th European Conference on Technology Enhanced Learning (EC-TEL 2018).

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin, 41*(1), 260–264.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: what is next for K-12? *Computers in Human Behavior, 41*, 51–61.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 1–15.

McGrew, K. S. (2009). CHC theory and the human cognitive abilities project: standing on the shoulders of the giants of psychometric intelligence research. *Intelligence, 37*(1), 1–10.

National Research Council (NRC). (2010). *Committee for the Workshops on Computational Thinking: Report of a workshop on the scope and nature of computational thinking*. Washington: National Academies Press.

National Research Council (NRC). (2011). *Committee for the Workshops on Computational Thinking: Report of a workshop of pedagogical aspects of computational thinking*. Washington: National Academies Press.

National Research Council (NRC) (2012). *A framework for K-12 science education: practices, crosscutting concepts, and core ideas*. National Academies Press.

Neumann, I., Fulmer, G. W., & Liang, L. L. (2013). Analyzing the FCI based on a force and motion learning progression. *Science Education Review Letters*, 2013:8-14.

NGSS Lead States. (2013). *Next generation science standards: for states, by states*. Washington: National Academies Press.

Prain, V., & Tytler, R. (2012). Learning through constructing representations in science: a framework of representational construction affordances. *International Journal of Science Education, 34*(17), 2751–2773.

Rasch, G. (1980). *Probabilistic models for some intelligence and attainment tests*. Chicago: University of Chicago Press.

Reiner, M., Slotta, J. D., Chi, M. T., & Resnick, L. B. (2000). Naive physics reasoning: a commitment to substance-based conceptions. *Cognition and Instruction, 18*(1), 1–34.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. B. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: a review and discussion. *Computer Science Education, 13*(2), 137–172.

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2016). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678–691.

Román-González, M., Pérez-González, J.-C., Moreno-León, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction, 18*, 47–58.

Rosenquist, M. L., & McDermott, L. C. (1987). A conceptual approach to teaching kinematics. *American Journal of Physics, 55*(5), 407–415.

Rutten, N., van Joolingen, W. R., & van der Veen, J. T. (2012). The learning effects of computer simulations in science education. *Computers & Education, 58*(1), 136–153.

Schwarz, C. V., & White, B. Y. (2005). Metamodeling knowledge: developing students' understanding of scientific modeling. *Cognition and Instruction, 23*(2), 165–205.

Schwarz, C. V., Reiser, B. J., Davis, E. A., Kenyon, L., Achér, A., Fortus, D., ... & Krajcik, J. (2009). Developing a learning progression for scientific modeling: making scientific modeling accessible and meaningful for learners. *Journal of Research in Science Teaching, 46*(6), 632-654.

Sengupta, P., Farris, A. V., & Wright, M. (2012). From agents to continuous change via aesthetics: learning mechanics with visual agent-based computational modeling. *Technology, Knowledge and Learning, 17*(1-2), 23–42.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: a theoretical framework. *Education and Information Technologies, 18*(2), 351–380.

Sequeira, M., & Leite, L. (1991). Alternate conceptions and history of science in physics teacher education. *Science Education, 75*(1), 45–56.

Sherin, B. L. (2001). A comparison of programming languages and algebraic notation as expressive languages for physics. *International Journal of Computers for Mathematical Learning, 6*(1), 1–61.

Sherin, B., diSessa, A. A., & Hammer, D. (1993). Dynaturtle revisited: learning physics through collaborative design of a computer model. *Interactive Learning Environments, 3*(2), 91–118.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science framework and NGSS: computational thinking in the science classroom. *Science Scope, 38*(3), 10–15.

Tasar, M. F. (2010). What part of the concept of acceleration is difficult to understand: the mathematics, the physics, or both? *ZDM, 42*(5), 469–482.

Tisue, S., & Wilensky, U. (2004, May). Netlogo: a simple environment for modeling complexity. *In International conference on complex systems, 21*, 16–21.

Tucker, A., Seehorn, D., Carey, S., Moix, D., Fuschetto, B., ... & Verno, A. (2011). CSTA K-12 computer science standards. CSTA Standards Task Force. Retrieved April 20, 2018, from http://www.education2020.ca/Content/K-12ModelCurrRevEd.pdf

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715–728.

Wagh, A., Cook-Whitt, K., & Wilensky, U. (2017). Bridging inquiry-based science and constructionism: exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching, 54*(5), 615–641.

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 199-208). ACM.

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education, 18*(1), 1–25.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: a dynamic systems approach to making sense of the world. *Journal of Science Education and Technology, 8*(1), 3–19.

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM, 57*(8), 24–28.

Wilkerson-Jerde, M. H., Gravel, B. E., & Macrander, C. A. (2015). Exploring shifts in middle school learners' modeling activity while generating drawings, animations, and computational simulations of molecular diffusion. *Journal of Science Education and Technology, 24*(2-3), 396–415.

Wu, H. K., & Puntambekar, S. (2012). Pedagogical affordances of multiple external representations in scientific processes. *Journal of Science Education and Technology, 21*(6), 754–767.

Yuruk, N., Beeth, M. E., & Andersen, C. (2009). Analyzing the effect of metaconceptual teaching practices on students' understanding of force and motion concepts. *Research in Science Education, 39*(4), 449–475.

Zur-Bargury, I., Parv, B., & Lanzberg, D. (2013). A nationwide exam as a tool for improving a new curriculum. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (pp. 267-272). ACM.