Increasing Student Self-Efficacy in Computational Thinking via STEM Outreach Programs

Russell Feldhausen
Kansas State University
Manhattan, Kansas
russfeld@ksu.edu

Joshua Levi Weese
Kansas State University
Manhattan, Kansas
weeser@ksu.edu

Nathan H. Bean
Kansas State University
Manhattan, Kansas
nhbean@ksu.edu

ABSTRACT

This paper describes our experiences developing and teaching two different interventions focused on computational thinking and computer science at a yearly STEM outreach program hosted by a local school district. We describe the creation of our lesson plans, how we worked with experienced and pre-service teachers alike to deliver the lessons, and how we assessed the effectiveness of each intervention. We will discuss our successes and failures, and provide information on our future plans to incorporate more formalized education theory, pedagogy, and research methodology in future years to further this project.

Based on our assessment results, we observed statistically significant gains in student self-efficacy with creating computer programs that perform a variety of operations. In addition, students reported a significantly higher understanding of how computer programming can be used in daily life. Our survey also highlighted differences in student self-efficacy between the two interventions, and we discuss possible sources for that result. We discuss observed results based on student groups with various backgrounds, previous STEM experiences, and socioeconomic status.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; *Computational thinking*; *Student assessment*;

KEYWORDS

pedagogy; self-efficacy; stem; outreach; k-12; middle school

ACM Reference Format:
Russell Feldhausen, Joshua Levi Weese, and Nathan H. Bean. 2018. Increasing Student Self-Efficacy in Computational Thinking via STEM Outreach Programs. In *SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, February 21–24, 2018, Baltimore , MD, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159593

1 INTRODUCTION

As the world becomes more reliant on technology, it is vital for students to learn relevant skills for navigating an increasingly computerized society. By introducing computational thinking (CT) to students in middle school, educators hope to have a positive impact on students' interest in computer science (CS) as well as their comfort level working with technology. Unfortunately, many K-12 curriculum standards do not include these skills. [6, 27] STEM (science, technology, engineering, mathematics) outreach programs are one effective way to introduce students to CS and CT. They provide freedom for educators to teach skills and subjects outside the normal curriculum standards, and create a fun and positive atmosphere for students to learn about possible future college majors and careers. [39]

A local school district invited us to assist with bringing CS to an ongoing STEM outreach program. We developed two intervention lesson plans to teach CT concepts to middle school students through this partnership. We collaborated with experienced educators and pre-service teachers to present the lessons to several cohorts of students, and collected assessment data on how the interventions impacted student self-efficacy with CT concepts.

2 RELATED WORK

In her seminal Viewpoint article, Wing described CT as a "fundamental skill for everyone." [40] She further expanded her definition in 2011, describing it as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." [41] This clearly implies that, while learning programming definitely teaches CT, there is a much broader realm of knowledge that fits this definition. Indeed, the Royal Society provided another take on CT, stating that it "is 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." [33] Using that definition, it is clear that CT can be applied to and taught within a variety of STEM fields.

Brennan and Resnick provided a more concrete definition for CT in terms of computational concepts, practices, and perspectives through their study of the Scratch [23] programming environment. [8] Similarly, Google also uses a list of concepts and skills to further define CT. [19] Weese and Feldhausen created a list of CT concepts and related CS principles based on those works, adapted below in Table 1, which informed the curriculum design in this work. [37]

Efforts to include CT in existing K-12 curricula are ongoing, but the progress is slow. Many states within the United States have voluntarily adopted the Common Core State Standards [26] and Next Generation Science Standards [35] within their K-12 schools. Reviews by Pokorny [27] and Bienkowski [6] individually state that while each set of standards include applicable mathematical

SIGCSE '18, February 21–24, 2018, Baltimore , MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159593

Table 1: CT Concepts and Related Computer Science Principles [37]

Abbr.	Description
ALG	**Algorithmic thinking** - sequence of steps that complete a task. Operators and expressions are also included
ABS	**Abstraction** - generalized representation of a complex problem, ignoring extraneous information
DEC	**Problem decomposition** - breaking a problem into smaller, more manageable parts that can be solved independently of each other
DAT	**Data** - collection, representation, and analysis of data [19]
PAR	**Parllelization** - simultaneous processing of a task [19]
CON	**Control flow** - directs an algorithm's steps when to complete
IAI	**Incremental and iterative** - building small parts of the program at each step instead of the whole program at once
TAD	**Testing and debugging** - performing intermediate testing and fixing problems while developing
QUE	**Questioning** - working to understand each part of the code instead of using code that is not understood well
USE	**Reuse and Remix** - making use of other people's work and resources to solve a problem

or CT skills, they fall short of deeply integrated CT. The Computer Science Teachers Association (CSTA) has also published a set of broad curricular standards for K-12 education, [3] based on the K-12 Computer Science Framework, [21] but our state has not adopted those standards. Wiebe et al. shared results of a statewide survey of student attitudes toward STEM fields, showing the importance of increased awareness and frequent exposure to STEM in encouraging students to consider future STEM careers. Since students may not be getting that exposure in the classroom due to limited support for CT and CS in K-12 standards, we must look to after-school outreach programs to fill the gap and connect students to STEM fields and practicing professionals. [39]

Research by Tai et al. published in the journal *Science* shows that, among students graduating with a 4-year college degree who were previously asked in 8th grade to list their possible future careers at age 30, those who expected to have a science-based career were "1.9 times more likely to earn a life science baccalaureate degree," and "3.4 times more likely to earn physical science and engineering degrees than students without similar expectations." [36] Similarly, that research showed that students with high mathematics scores were also much more likely to graduate with physical science and engineering degrees. This clearly demonstrates that early interest in science and mathematics, two major parts of STEM, has a large impact in future career choices.

In 2014, a group of CS education researchers concluded that "[t]he greatest need in providing computing education in schools is to provide more teachers." [13] Bean et al. described a program training pre-service teachers in CT, with the goal of encouraging them to integrate more CT and programming into their future classrooms. [5] The STEM program described below similarly involves pre-service teachers to achieve the same goal.

There are many methods to assess CT in students. One possibility is to use evidence-centered design to create assessments and quantitatively measure student learning outcomes. [32] However, due to the informal and fun nature of STEM outreach programs, this approach was deemed unreasonable.

Another approach is to measure student self-efficacy, described by Bandura as "how people judge their capabilities" and the effect that judgement has on their own "motivation and behavior." [4] Ramalingam et al. studied the self-efficacy of students before and after a programming class, and compared that to the measured mental models of the participants. They found that not only do most students experience significant gains in self-efficacy, but student self-efficacy itself was a significant predictor of student performance, along with the student's own mental model. [29] Lishinski et al. furthered this work by studying how self-efficacy and "self-regulated learning" aspects relate to course performance over time. They confirmed that self-efficacy is the strongest predictor of student performance in classes when compared to other predictors such as student goals and metacognitive strategies. [24]

3 INTERVENTION DESIGN

We developed two different interventions for this STEM outreach program, each for a different age group and with a different theme. Each intervention included four 3-hour sessions each week, and was presented to four different groups of 12 to 18 students each summer. Below we discuss the design of each intervention, focusing on our own intentions and thoughts when developing the interventions. More details on the interventions can be found in Weese and Feldhausen [37] and online. [18]

3.1 Saving the Martian

Saving the Martian is an intervention for 5th and 6th grade students that focuses on introducing CT through unplugged activities and scaffolded examples using the Scratch programming environment. We found inspiration in the book *The Martian*, by Andy Weir, [38] by looking for scenarios that could be generalized into activities for the class, such as the protagonist's struggle to grow food and communicate. We could then use the book to provide additional framing for the activities along with the field of space travel, hoping to engage students with an interesting theme. The class focused on four primary areas: programming, algorithms, data, and artificial intelligence. Each area included unplugged activities, scaffolded programming exercises, and discussion of how each related to CS and other topics.

As we introduced programming via the Scratch environment, we related programming terms to the familiar concept of a school play. We also used activities to create simple programs, clearly describing the thought processes a programmer would use to complete the task, while giving students ample time to experiment and build their own knowledge. This process follows the paradigms of educational

constructivism and constructionism popularized by Piaget and Papert. [2] To introduce algorithms, we used sorting algorithms in both unplugged and programming activities, coupled with a discussion of simple algorithmic analysis to compare different algorithms for the same task. We also created simple computer simulations to manipulate data variables, showing how chemical reactions would take place in order to grow food on Mars. When dealing with data, students learned both binary and hexadecimal number systems, as well as ASCII, all to build a program to translate communicated data. We also included some ambiguous messages in our activity, prompting a discussion of reliable communication methods. Finally, our activities for artificial intelligence included a discussion of the Turing test, an unplugged activity introducing artificial neural networks, and programming projects to create perceptron-based AIs for video games and navigation.

Throughout the week, we focused on blending both unplugged and programming activities to show how CT and CS work in different fields. We also provided outside information and expansive framing whenever possible to link our activities and CS in general to the broader world. By showing how CS can have an impact in everyone's daily lives, we hope to encourage students to consider CS as a future career.

3.2 Mighty Micro Controllers

Mighty Micro Controllers (MMC) is an intervention for 7th through 9th grade students, focused on teaching CT through programming Arduino Uno micro controllers using Scratch, the Arduino IDE, and a text-based language. Our goal for MMC was to use physical computing as a means to engage students with CS and CT, which has been shown to be successful. [25, 28] The format of this class included guided examples creating simple circuits and programs, followed by problem-driven exploration to build programming, electrical, and CT skills. We also used pair programming extensively to further aid student success. [7]

We taught students the basic principles of electricity, circuits, and signals. As these concepts are new to many students, it can be overwhelming at first, so we drew upon unplugged examples to manage the challenge. For example, we adopted Kuphalt's physical illustration of electricity using marbles flowing in a hula-hoop. [22] Similarly, students were challenged to rank resistors from weakest to strongest through a hands-on experiment by plugging the resistors into a blinking LED circuit, discovering that the weaker resistors let the LED shine bright, while the strongest let nearly no light out at all.

Each successive activity had an increasing focus on CT. These were scaffolded to emphasize the CT and software side of micro controllers, rather than the physical implementation. Students were given detailed circuit diagrams for each activity, and guided through the steps to enable the required pins on the Arduino. This left students to program their own ideas into each activity. With most of the circuits done with LEDs, groups were encouraged to show originality in patterns and light shows. However, we found that the physical devices can be distracting. When given time to explore after finishing a guided activity, students often spent time focusing on getting the hardware to work instead of the code. This issue arose in debugging as well. When an LED would not light up, students often focused on the circuit, though the cause was typically a misplaced block of code.

Students were also introduced to sensors. One final guided activity used ultrasonic sensors within the Arduino IDE, since Scratch was not able to accurately work with these sensors. We introduced students to a text-based programming language, showing how it directly mirrored the structure of a Scratch program. This gave context for students to connect how the blocks they had been using corresponded to a real-world application. Students were then given ample time to complete a final group project for the class. Some students truly innovated in creating their own projects by going beyond the examples given in class, while others simply continued projects done throughout the week. This is one of the fundamental principles of CT; however, we saw many groups that relied too heavily on reusing their old material and programs without sufficient changes and innovation, an area we would like to improve in future implementations.

4 STEM SUMMER INSTITUTE

The *STEM Summer Institute*, [1] hosted by the Manhattan-Ogden USD 383 school district and funded by a grant though the United States Department of Defense, "is designed to raise student achievement levels and increase enrollments in science, technology, engineering and math, or STEM, careers" [31] for students in 5th grade through 9th grade. Teachers from the school district, pre-service teachers from the Kansas State University College of Education, and subject-matter experts are grouped together to develop and teach each class during the camp.

We filled the role of subject-matter experts, and were paired with teachers from the school district interested in our field. We worked collaboratively with the teachers to develop each intervention plan in the spring. Typically, we developed the lessons, and the teachers helped us make sure they were appropriate for the younger students. They also provided feedback on how to link our lessons with the school's curricula. At the start of each summer camp, we were joined by one to three pre-service teachers for each intervention. They were primarily there to gain in-class teaching experience while also getting exposure to STEM disciplines to be used in future classrooms. Pre-service teachers were required to teach at least one lesson during the last week to demonstrate their capability working with the material.

During the first two weeks, we would lead the class through most of the activities, with the teachers observing and assisting as needed. The teachers would also frequently add more information or clarify something if needed, helping us to refine our lesson plans to fit the students. For the last two weeks, the teachers would progressively lead more and more of the class, with us offering advice or corrections as needed.

5 ASSESSMENT

As discussed in the related work, we were unable to perform direct knowledge assessments since it would not be appropriate for the program. Instead, we focused on measuring student self-efficacy in problem solving and computer programming. In doing so, we hope to show that our interventions increase student interest in CS and comfort level with CT concepts, thereby making them more likely

Table 2: Self-Efficacy Survey Questions and Related CT Skill [37]

	When solving a problem I...	
1	create a list of steps to solve it	ALG - Algorithms
2	use math	ALG - Algorithms
3	try to simplify the problem by ignoring details that are not needed [41]	ABS - Abstraction
4	look for patterns in the problem	ABS - Abstraction
5	break the problem into smaller parts	DEC - Problem Decomposition
6	work with others to solve different parts of the problem at the same time	PAR - Parallelization
7	look how information can be collected, stored, and analyzed to help solve the problem	DAT - Data
8	create a solution where steps can be repeated [30]	CON - Control Flow
9	create a solution where some steps are done only in certain situations [32]	CON - Control Flow
	I can write a computer program which...	
10	runs a step-by-step sequence of commands	ALG - Algorithms
11	does math operations like addition and subtraction	ALG - Algorithms
12	uses loops to repeat commands	CON - Control Flow
13	responds to events like pressing a key on the keyboard	CON - Control Flow
14	only runs specific commands when a specific condition is met	CON - Control Flow
15	does more than one thing at the same time	PAR - Parallelization
16	uses messages to talk with different parts of the program	PAR - Parallelization
17	can store, update, and retrieve values	DAT - Data
18	uses custom blocks	ABS - Abstraction
	When creating a computer program I...	
19	make improvements one step at a time and work new ideas in as I have them	IAI - Incremental and Iterative
20	run my program frequently to make sure it does what I want and fix any problems I find	TAD - Testing and Debugging
21	share my programs with others and look at others' programs for ideas	USE - Reuse and Remix
22	break my program into multiple parts to carry out different actions	DEC - Problem Decomposition
	Impact	
23	I understand how computer programming can be used in my daily life	QUE - Questioning

to choose a STEM major or career in the future. The assessments used for this program are detailed in Weese and Feldhausen, [37] and related survey questions are included in Table 2.

In addition, we asked students about their previous experiences with various programming environments, other STEM programs, and simple questions measuring a student's socioeconomic status. [14, 17, 20] Those questions help us further categorize our students to observe any relevant differences between groups.

6 RESULTS

During 2016 and 2017, we taught each intervention to a total of 214 students across 16 groups. Of those, we collected valid pre- and post-surveys from 184 students. Students were excluded from the survey for failing to complete either a pre- or post-survey, or by opting-out of the research. We failed to administer the post-survey to one Saving the Martian intervention due to time constraints, accounting for a majority of students excluded from the results. We did not observe significant differences between each year's data, and therefore combined them. Our combined results for Cohen's d effect size [10] for both years are detailed in Table 3. Effect sizes are marked as small, medium, or large effect following Cohen's recommended reference points, [10] with significant results shaded based on the associated p-values. The results are divided into the following groupings:

- **Mars** - Saving The Martian
- **MMC** - Mighty Micro Controllers
- **Male** - Male-identifying students
- **Female** - Female-identifying students
- **NoS** - Haven't previously attended a STEM event
- **STEM** - Have previously attended a STEM event
- **SB** - Have previously attended STARBASE
- **>$10** - More than $10 in weekly spending money
- **≤$10** - $10 or less in weekly spending money
- **Cell** - Have a cell phone of their own
- **NoC** - No cell phone of their own
- **MS** - At least one parent has MS degree or higher
- **NoMS** - No parent has MS degree or higher

Our data shows that we clearly achieved gains in student self-efficacy related to CT skills when framed as writing computer programs, but not when those skills were framed within problem solving. This supports the claim that learning CT is not the same as learning problem solving, and that simply teaching a student about programming will not necessarily make them better problem solvers. We found significant gains in student self-efficacy in a

Table 3: Effect Size ($post - pre/stdev$) [10] for Problem Solving (PS) and CT Concepts. Italicized values indicate a small (≥.2) effect, bold values indicate a medium (≥.5) effect, and bold italicized values indicate a large (≥.8) effect. Shaded values indicate a p-value ≤.01. Columns marked with an asterisk include data from 2017 only

Skill	All	Mars	MMC	Male	Female	NoS	STEM	SB	>$10*	≤$10*	Cell*	NoC*	MS*	NoMS*
PS - ALG	0.15	*0.28*	0.05	0.12	*0.25*	0.19	0.15	*0.21*	0.14	0.16	0.13	*0.22*	0.15	-0.04
PS - ABS	0.01	0.09	-0.07	0.01	0.00	0.02	0.00	0.10	0.02	-0.08	-0.02	-0.15	-0.02	-0.08
PS - DEC	0.12	*0.23*	0.01	0.08	0.17	0.00	0.15	0.18	*0.36*	-0.11	0.03	-0.09	0.00	0.13
PS - PAR	0.05	0.16	-0.05	0.04	0.10	0.00	0.07	0.11	-0.05	-0.02	0.13	-0.44	-0.33	*0.24*
PS - DAT	0.11	0.04	0.17	0.04	*0.21*	-0.02	0.15	*0.32*	0.15	-0.03	0.03	-0.04	-0.09	0.00
PS - CON	0.15	*0.21*	0.10	0.12	*0.22*	0.13	0.16	*0.26*	0.05	0.04	0.07	-0.05	-0.11	0.04
CT - ALG	*0.41*	**0.56**	*0.28*	*0.45*	*0.33*	*0.47*	*0.39*	**0.58**	*0.39*	*0.39*	*0.39*	*0.40*	*0.42*	**0.65**
CT - ABS	**0.66**	*0.48*	***0.84***	**0.71**	**0.61**	*0.48*	**0.71**	***0.80***	***0.89***	**0.77**	**0.76**	***0.93***	**0.66**	***1.52***
CT - DEC	*0.39*	*0.48*	*0.29*	*0.30*	**0.55**	*0.41*	*0.38*	**0.55**	*0.40*	*0.40*	*0.44*	*0.30*	*0.40*	**0.70**
CT - PAR	*0.42*	*0.48*	*0.37*	*0.45*	*0.36*	*0.44*	*0.42*	**0.57**	*0.47*	*0.44*	*0.48*	*0.35*	*0.37*	**0.56**
CT - DAT	**0.66**	**0.74**	**0.58**	**0.69**	**0.61**	**0.57**	**0.69**	***0.86***	*0.50*	**0.76**	**0.65**	**0.82**	*0.45*	**0.57**
CT - CON	*0.29*	*0.33*	*0.26*	*0.28*	*0.34*	*0.27*	*0.30*	*0.42*	*0.38*	*0.35*	*0.37*	*0.34*	*0.47*	*0.33*
CT - IAI	*0.29*	0.19	*0.39*	*0.28*	*0.27*	*0.34*	*0.28*	*0.36*	**0.60**	*0.28*	*0.30*	**0.51**	*0.34*	**0.60**
CT - TAD	*0.26*	*0.27*	*0.26*	*0.26*	*0.27*	*0.30*	*0.25*	*0.34*	*0.23*	*0.47*	*0.38*	*0.49*	*0.25*	**0.61**
CT - USE	0.19	0.18	*0.20*	*0.26*	0.05	*0.33*	0.15	0.15	-0.23	*0.28*	0.12	*0.22*	*0.28*	-0.12
CT - QUE	*0.41*	**0.54**	*0.30*	**0.51**	*0.26*	**0.54**	*0.37*	**0.61**	*0.39*	*0.50*	*0.44*	**0.59**	*0.39*	**0.61**
# Students	184	90	94	121	61	42	142	51	21	58	58	21	41	9
Pre Mean	3.68	3.76	3.61	3.70	3.64	3.65	3.69	3.73	3.72	3.64	3.68	3.59	3.70	3.69
Post Mean	4.03	4.17	3.89	4.05	3.99	4.01	4.03	4.21	4.06	3.98	4.04	3.89	4.01	4.08

variety of CT skills, with each skill except reuse and remix (USE) showing at least a small effect size, with appropriately significant p-values. Our largest effect sizes were observed in the areas of abstraction (ABS), data (DAT) and questioning (QUE). Students in the Mars group showed higher gains in algorithms (ALG) than any other group, primarily due to that class's focus on algorithms. We observed lower effect sizes for students in MMC, which we attribute to the increased cognitive load of learning circuits along with programming. Males and females were largely similar, though female students had a larger effect size in the area of problem decomposition (DEC) while males were higher in the overall questioning and impact (QUE) area.

Students in the STEM group had larger effect sizes than students in the NoS group in most areas, which we attribute to repeated positive exposure to STEM; students who have already gained some experience in STEM may be more likely to engage with the material. Similarly, we observed medium and high effect sizes in students who have previously attended STARBASE, a structured STEM program for 5th graders in the area. [34] We did not observe significant differences between student groups based on self-reported socioeconomic information. While the data indicates that students with a lower socioeconomic status tend to experience larger effect sizes in some areas, the small population size makes it difficult to draw any definitive conclusions. We hope to continue this research in future years to get additional data in this area.

Based on experiences throughout the camp, we observed a significant increase in the ability of partnered teachers to engage with CT concepts and instruct students in a variety of technical activities. In most cases, we were able to completely allow them to lead the entire class for the last week each summer, only adding a few comments when students had questions. We found this to be a very valuable way to bring CT to educators outside CS. In addition, working with experienced teachers allowed us to improve our own skills working with younger students. We learned techniques for maintaining student attention, encouraging students to discuss among themselves, and how to link our own activities back to things they have already covered in their classrooms. They also provided valuable comments and suggestions for our intervention designs, allowing us to refine them each week based on their feedback. We found this to be a very positive experience for everyone involved, and believe it is an effective method to improve our own teaching and outreach efforts.

7 CONCLUSIONS AND FUTURE WORK

We developed two STEM interventions targeted at 5th through 9th grade students, and partnered with teachers from a local school district as well as pre-service teachers from our university to teach these interventions during a summer STEM camp in 2016 and 2017. We observed significant gains in student self-efficacy with many CT skills framed within computer programming. We did not observe similar gains in CT skills framed within problem solving. Additionally, we found that working with practicing and pre-service teachers was a positive experience for everyone involved, and we feel that it greatly improved our intervention design and teaching skills. Finally, we gained valuable experience creating STEM intervention curricula and performing basic research, informing

our future efforts to formalize our curriculum design and research methods.

We plan to continue this project using more formalized research methods. First, we would like to revisit each intervention's design and add techniques from educational theory and pedagogy, particularly cognitive apprenticeship [9, 11, 12] and expansive framing, [15, 16] both of which could have positive impacts on student learning and closely match our existing design. We also plan to continue to refine our self-efficacy assessment by combining it with knowledge-based assessments to confirm that students are indeed learning the material while also becoming more confident in the field. While we cannot directly assess student learning through exams, we hope to make use of student trivia games with evidence-based questions to drive that research. Finally, we plan on expanding the scope of our research by using activities from each intervention during other outreach events while collecting data from participants. This will help expand our pool of research and give us insight into how other student populations and events affect our results.

REFERENCES

[1] Manhattan-Ogden USD 383. 2017. STEM Summer Institute. (2017). Retrieved July 13, 2017 from http://www.usd383.org/manhattan-ogden/district-office-/teaching-and-learning/stem/stem-summer-institute

[2] Edith Ackermann. 2001. Piaget's Constructivism, Papert's Constructionism: What's the Difference. In *Constructivism: Uses and Perspectives in Education*. 85–94.

[3] Computer Science Teachers Association. 2017. CSTA K-12 Computer Science Standards, Revised 2017. (2017). Retrieved July 31, 2017 from http://www.csteachers.org/standards

[4] Albert Bandura. 1982. Self-efficacy Mechanism in Human Agency. *American Psychologist* 37, 2 (02 1982), 122–147.

[5] Nathan Bean, Joshua Weese, Russell Feldhausen, and Richard Scott Bell. 2015. Starting from Scratch: Developing a Pre-service Teacher Training Program in Computational Thinking. In *2015 IEEE Frontiers in Education Conference (FIE)*. 1–8. https://doi.org/10.1109/FIE.2015.7344237

[6] Marie Bienkowski. 2016. Deepening Learning in High School Computer Science Through Practices in the NGSS. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 694–694. https://doi.org/10.1145/2839509.2850557

[7] Grant Braught, Tim Wahls, and L. Marlin Eby. 2011. The Case for Pair Programming in the Computer Science Classroom. *Trans. Comput. Educ.* 11, 1, Article 2 (Feb. 2011), 21 pages. https://doi.org/10.1145/1921607.1921609

[8] Karen Brennan and Mitchel Resnick. 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*.

[9] John Seely Brown, Allan Collins, and Paul Duguid. 1989. Situated Cognition and the Culture of Learning. *Educational Researcher* 18, 1 (1989), 32–42. https://doi.org/10.3102/0013189X018001032

[10] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences - Second Edition*. Lawrence Erlbaum Associates.

[11] Allan Collins. 1988. Cognitive Apprenticeship and Instructional Technology. Technical Report. *BBN Systems and Technologies Corporation* (1988).

[12] Allan Collins, John Seely Brown, and Ann Holum. 1991. Cognitive Apprenticeship: Making Thinking Visible. *American Educator* 15, 3 (1991), 6 – 11, 38–46.

[13] Stephen Cooper, Linda Bookey, and Peter GruenBaum. 2014. *Future Directions in Computing Education Summit Part One: Important Computing Education Research Questions* . Technical Report CS-TR-14-0108-SC. Stanford University. http://ilpubs.stanford.edu:8090/1117/

[14] Candace E. Currie, Rob A. Elton, Joanna Todd, and Stephen Platt. 1997. Indicators of socioeconomic status for adolescents: the WHO Health Behaviour in School-aged Children Survey. *Health Education Research* 12, 3 (1997), 385–397. https://doi.org/10.1093/her/12.3.385

[15] Randi A. Engle, Diane P. Lam, Xenia S. Meyer, and Sarah E. Nix. 2012. How Does Expansive Framing Promote Transfer? Several Proposed Explanations and a Research Agenda for Investigating Them. *Educational Psychologist* 47, 3 (2012), 215–231. https://doi.org/10.1080/00461520.2012.695678

[16] Randi A. Engle, Phi D. Nguyen, and Adam Mendelson. 2011. The Influence of Framing on Transfer: Initial Evidence from a Tutoring Experiment. *Instructional Science* 39, 5 (01 Sep 2011), 603–628. https://doi.org/10.1007/s11251-010-9145-2

[17] Doris R. Entwisle and Nan Marie Astone. 1994. Some Practical Guidelines for Measuring Youth's Race/Ethnicity and Socioeconomic Status. *Child Development* 65, 6 (1994), 1521–1540. http://www.jstor.org/stable/1131278

[18] Russell Feldhausen. 2017. Curriculum Resources. (2017). Retrieved August 31, 2017 from http://people.cs.ksu.edu/~russfeld/curriculum/

[19] Google. 2017. Computational Thinking Concepts Guide. (2017). Retrieved August 24, 2017 from https://docs.google.com/document/d/1i0wg-BMG3TdwsShAyH_0Z1xpFnpVcMvpYJceHGWex_c/edit

[20] Robert M. Hauser. 1994. Measuring Socioeconomic Status in Studies of Child Development. *Child Development* 65, 6 (1994), 1541–1545. http://www.jstor.org/stable/1131279

[21] K-12 Computer Science Framework. 2016. K-12 Computer Science Framework. (2016). http://www.k12cs.org

[22] Tony R. Kuphaldt. 2014. Lessons in Electric Circuits. (2014). Retrieved August 29, 2017 from https://www.allaboutcircuits.com/textbook/

[23] MIT Media Lab. 2017. Scratch - Imagine, Program, Share. (2017). Retrieved July 21, 2017 from https://scratch.mit.edu/

[24] Alex Lishinski, Aman Yadav, Jon Good, and Richard Enbody. 2016. Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 211–220. https://doi.org/10.1145/2960310.2960329

[25] Raina Mason and Graham Cooper. 2013. Mindstorms robots and the application of cognitive load theory in introductory programming. *Computer Science Education* 23, 4 (2013), 296–314. https://doi.org/10.1080/08993408.2013.847152

[26] Council of Chief State School Officers National Governors Association Center for Best Practices. 2010. Common Core State Standards. (2010). Retrieved August 14, 2017 from http://www.corestandards.org/

[27] Kian L. Pokorny. 2013. What Will They Know? Standards in the High School Computer Science Curriculum. *J. Comput. Sci. Coll.* 28, 5 (May 2013), 218–225. http://dl.acm.org/citation.cfm?id=2458569.2458616

[28] Mareen Przybylla and Ralf Romeike. 2014. Physical computing and its scope-towards a constructionist computer science curriculum with physical computing. *Informatics in Education* 13, 2 (2014), 225.

[29] Vennila Ramalingam, Deborah LaBelle, and Susan Wiedenbeck. 2004. Self-efficacy and Mental Models in Learning to Program. *SIGCSE Bull.* 36, 3 (June 2004), 171–175. https://doi.org/10.1145/1026487.1008042

[30] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[31] Patrice Scott. 2012. Summer STEM Institute is an academic adventure. (Jun 2012). Retrieved July 12, 2017 from https://www.k-state.edu/today/announcement.php?id=4050

[32] Eric Snow, Geneva Haertel, Dennis Fulkerson, M Feng, and P Nichols. 2010. Leveraging evidence-centered assessment design in large-scale and formative assessment practices. In *Proceedings of the 2010 Annual Meeting of the National Council on Measurement in Education (NCME)*.

[33] The Royal Society. 2012. Shut down or restart? The way forward for computing in UK schools. (2012). Retrieved August 23, 2017 from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

[34] Kansas STARBASE. 2017. Kansas STARBASE. (2017). Retrieved August 31, 2017 from http://www.kansasstarbase.org/

[35] NGSS Lead States. 2013. Next Generation Science Standards. (2013). Retrieved August 14, 2017 from https://www.nextgenscience.org/

[36] Robert H. Tai, Christine Qi Liu, Adam V. Maltese, and Xitao Fan. 2006. Planning Early for Careers in Science. *Science* 312, 5777 (2006), 1143–1144. https://doi.org/10.1126/science.1128690

[37] Joshua Levi Weese and Russell Feldhausen. 2017. STEM Outreach: Assessing Computational Thinking and Problem Solving. In *2017 ASEE Annual Conference & Exposition*. ASEE Conferences, Columbus, Ohio. https://peer.asee.org/28845

[38] Andy Weir. 2014. *The Martian*. Crown.

[39] Eric N. Wiebe, Malinda Faber, Jeni Corn, Tracey Louise Collins, Alana Unfried, and LaTricia Townsend. 2013. A Large-scale Survey of K-12 Students about STEM: Implications for Engineering Curriculum Development and Outreach Efforts (Research to Practice) . In *2013 ASEE Annual Conference & Exposition*. ASEE Conferences, Atlanta, Georgia. https://peer.asee.org/19073

[40] Jeannette M. Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. https://doi.org/10.1145/1118178.1118215

[41] Jeannette M. Wing. 2011. Research notebook: Computational thinking - What and why? (2011). Retrieved July 21, 2017 from http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why