A COMPARISON OF COMPUTER-BASED AND ROBOTIC PROGRAMMING
INSTRUCTION:
IMPACT OF SCRATCH VERSUS COZMO ON MIDDLE SCHOOL STUDENTS'
COMPUTATIONAL THINKING, SPATIAL SKILLS, COMPETENCY BELIEFS,
AND ENGAGEMENT

A dissertation submitted to the
Kent State University College
of Education, Health, and Human Services
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

By

Shannon Marshall Smith

August 2019

ProQuest Number: 27602977

ProQuest 27602977

A dissertation written by

Shannon Marshall Smith

B.S., Ohio University, 1997

M.A., Muskingum University, 2004

Ph.D., Kent State University, 2019

Approved by

_____, Director, Doctoral Dissertation Committee
Elena Novak

_____, Member, Doctoral Dissertation Committee
Chia-Ling Kuo

_____, Member, Doctoral Dissertation Committee
Jason Schenker

Accepted by

_____, Director, School of Lifespan Development and
Mary Dellmann-Jenkins   Educational Services

_____, Dean, College of Education, Health, and Human
James Hannon   Services

iii

SMITH, SHANNON MARSHALL, Ph.D., August 2019

A COMPARISON OF COMPUTER-BASED AND ROBOTIC PROGRAMMING
INSTRUCTION: IMPACT OF SCRATCH VERSUS COZMO ON MIDDLE SCHOOL
STUDENTS' COMPUTATIONAL THINKING, SPATIAL SKILLS, COMPETENCY
BELIEFS, AND ENGAGEMENT (249 pp.)

Director of Dissertation: Elena Novak, Ph.D.

ABSTRACT

The purpose of this study was to examine the effects of coding activities supported by the artificially intelligent, animated emotional-educational robot Cozmo on middle school students' computational thinking, spatial skills, competency beliefs, and engagement compared to the more traditional computer-based program of Scratch. Two versions of the coding curriculum unit were developed. Both versions shared the same content and instructional features, but differed in the code blocks used in the Scratch and Cozmo programs. Two intact seventh grade classes at a public middle school in the Midwest participated in the study during their regularly scheduled Technology course. One class received the Scratch coding curriculum ($N = 21$), and the other class received the robotics coding curriculum ($N = 22$).

Results revealed non-significant differences in computational thinking, mental rotation skills and competency beliefs among the Scratch and Cozmo interventions. However, students found Cozmo to be more engaging than Scratch. Both interventions

significantly improved students' computational thinking skills, mental rotation skills, and competency beliefs from pre- to post-test.

This study contributes to the scarce literature on programming education in a public school setting with a diverse group of students. The positive gains in both the cognitive and affective domains of learning found in this study are encouraging and can serve as a point of reference for researchers, curriculum designers, and educators with the desire to introduce students to programming.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my dissertation director Dr. Elena Novak for guiding, challenging, and supporting me throughout the entire dissertation process. I gratefully acknowledge my dissertation committee members Dr. Chia-Ling Kuo and Dr. Jason Schenker for their invaluable advice and feedback, my graduate faculty representative Dr. Jong-Hoon Kim for his helpful suggestions, and researchers Anthony Shreffler and Fred Nyangaresi for their assistance and reassurance during this journey. I would also like to thank the Graduate Student Senate for funding and the students and staff at New Philadelphia City Schools for making this study possible.

The completion of this dissertation would not have happened without the unconditional love and support of my family. My parents have always been models of the perfect blend of love, humor, fun, and hard work. Doug is still a beacon capable of brightening the darkest day. My children are my greatest inspiration and their love, patience, and support have sustained me throughout this process; from the cups of tea, to the notes and hugs that kept me going through every challenge. The Friday lunches and unending support and laughs made possible by my friend Debbie have also been instrumental throughout my studies and the dissertation process. Words cannot adequately express my love and thanks for the blessing of my family and friends. This dissertation is dedicated to them. In the words of 1 Corinthians 13:13, "And now these three remain: faith, hope, and love. But the greatest of these is love."

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

xi

CHAPTER I

INTRODUCTION

Background

There is wide acknowledgement of the need to broaden participation in Science, Technology, Engineering, and Mathematics (STEM) fields (Donnelly, 2013). According to the U.S. Bureau of Labor Statistics (2015), seven out of the ten largest STEM occupations were computer related, and employment in computer occupations is projected to increase by 12.5 percent from 2014 to 2024 (Fayer, Lacey, & Watson, 2017). This growth is expected to result in nearly half a million new computer-related jobs, far more than any other STEM group. The computer occupational group alone is projected to yield over 1 million job openings from 2014 to 2024. These statistics demonstrate a demand for computer skills, yet there is a lack of access to computer science courses for under-represented minorities (African Americans, Latinos, and Native Americans) (Margolis, Estrella, Goode, Holme, & Nao, 2008) and a significant gender gap in students enrolling in advanced placement computer science courses (CollegeBoard, 2017). According to the authors of the report titled *Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age*:

> At a time when computing is driving job growth and new scientific discovery, it is unacceptable that roughly two-thirds of the entire country has few computer science standards for secondary school education, K-8 computer science standards are deeply confused, few states count computer science as a core academic

1

subject for graduation, and computer science teacher certification is deeply flawed. (Wilson, Sudol, Stephenson, & Stehlik, 2010, p. 8)

Computer programming skills contribute to the development of other higher level skills like problem-solving, inferencing, and creative thinking (Blakemore, 2017; Brichacek, 2014; Fesakis, Gouli, & Mavroudi, 2013; Fesakis & Serafeim, 2009; Kay & Knaack, 2005) and a number of countries are beginning to teach computer and programming courses. In Turkey, the Information Technologies and Software courses used to be taught as elective courses, but have been compulsory for secondary schools since 2012 (Gökçearslan, Günbatar, & Kukul, 2017). Israel has a mandatory high school computer science curriculum and countries including Russia, South Africa, New Zealand, and Australia have made room for computer science in the K-12 curriculum (Grover & Pea, 2013, p. 40). The United Kingdom changed its National Curriculum for England with revisions to its K-12 computing programs of study. The new curriculum suggests that "a high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world" (DfE, 2013, p. 230). The teaching of computational thinking skills has also been included as an expectation in the Next Generation Science Standards in the United States (NGSS, 2017), as an interwoven theme in the K-12 Computer Science Framework (ACM, Code.org, CSTA, Cyber Innovation Center, & National Math and Science Initiative, 2016), and as one of the standards for students by the International Society for Technology in Education (ISTE) (Sandars, Van Oss, & McGeary, 2016). Some advocate that computational thinking is at the core of all of the STEM disciplines (Henderson, Cortina, Hazzan, & Wing, 2007;

Weintrop et al., 2016), that it is applicable and useful in everyday life (not just for computer science majors) (Lye & Koh, 2014), that it can be transferred to various types of problems that do not directly involve programming tasks (Wing, 2008), and that it is increasingly seen as an essential skill to create rather than just consume technology (Resnick et al., 2009). So what exactly is computational thinking?

Computational thinking

Finding an exact definition for computational thinking can be difficult (S. Grover & Pea, 2013; Shute, Sun, & Asbell-Clarke, 2017). Wing (2010) explained that "computational thinking is 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" (paragraph 1). Shute, Sun, & Asbell-Clarke (2017) built upon previous work to develop a working definition of computational thinking: "The conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts" (p. 151). Both of these definitions point toward a way of thinking and acting that can be exhibited through the use of particular skills.

The assertion has been made that computational thinking is considered a skill that everyone, not just computer scientists, should have (Korkmaz, Çakir, & Özden, 2017; Wing, 2006, 2008, 2010). This has prompted researchers and educators to focus their attention on computer programming (Ke, 2014; Uysal & Yalin, 2012), but it is considered

difficult by students and teachers alike (Armoni, 2011; Caspersen & Kölling, 2009; Gökçearslan & Alper, 2015; Nilsen & Larsen, 2011; Shadiev et al., 2014). Askar and Davenport (2009) observed that students had low performance in computer programming courses. Researchers have pointed out a multitude of reasons for this, including difficulties writing programs on a program-compiler and programming instruction that uses uninteresting activities (Resnick et al., 2009) as well as traditional classrooms that emphasize and reward solitary programmers (Malcom et al., 2005; Margolis & Fisher, 2002). Overcoming these challenges is a noteworthy goal because when children learn a programming language, they are not just "learning to code, they are coding to learn" (Resnick, 2013, p.5). Programming facilitates systematic problem solving, self-expression, and the challenge of learning powerful new ideas (Sullivan & Bers, 2016, p. 5).

Scratch

Visual programming languages like Scratch, Alice, and App Inventor were developed with the goal of making it easier for younger students to learn the basics of computer programming. Papert (1980) argued that programming languages should have a "low floor" (easy to begin), a "high ceiling" (able to create more complex projects over time), and "wide walls" (supports many different types of projects to appeal to people with different interests). The language of Scratch was based on this concept and it has been used to generate interest and engagement in computer science, especially for girls and ethnic groups that have been historically underrepresented in computing fields (Malcom et al., 2005; Maloney, Peppler, Kafai, Resnick, & Rusk, 2008; Margolis &

Fisher, 2002; Monroy-Hernández & Resnick, 2015). Scratch uses a collection of graphical programming blocks that can be snapped together to create programs. The blocks are shaped to fit together only in ways that make syntactic sense so that novices do not have to learn the complex syntax of punctuation required by traditional programming languages.

Scratch has been one of the more widely studied tools for teaching programming in both formal and informal settings. A growing number of K-12 schools worldwide and some universities, including Harvard, use Scratch as a first step in learning programming (Brennan, Balch, & Chung, 2011). A literature review of empirical studies on the Maker Movement revealed that all of the studies used some type of digital material to support making activities, and Scratch was the most widely used element (Papavlasopoulou, Giannakos, & Jaccheri, 2017). The Maker Movement refers to spaces (in and/or outside of a formal classroom environment) that allow people from all walks of life to express their creativity by making digital or tangible objects (Papavlasopoulou et al., 2017, p. 58).

Scratch has also been used to develop various computational frameworks. For instance, a Progression of Early Computational Thinking (PECT) Model was developed based on design patterns from programs written in Scratch to assess computational thinking in the primary grades (first through sixth) (Seiter & Foreman, 2013). Another computational thinking framework developed in the context of Scratch uses computational concepts, practices and perspectives as its key dimensions (Brennan & Resnick, 2012). The researchers claimed that the seven computational concepts that they identified were found to be highly useful in a wide range of Scratch projects and

transferable to other programming and non-programming contexts. There are even computational thinking courses that have been developed using Scratch. Grover, Pea, and Cooper (2015) claimed that seventh and eighth grade students who took a 7-week computational thinking course using Scratch were able to transfer their skills from Scratch to a text-based programming context.

While Scratch has many reported benefits, there are noted issues as well. A number of studies have found that certain computational concepts were rarely or never used in students' Scratch projects (Aivaloglou & Hermans, 2016; Denner, Werner, & Ortiz, 2012; Fields, Kafai, & Giang, 2017). One study claimed that the Scratch projects analyzed at a Computer Clubhouse demonstrated learning of key programming concepts even though the youth who created them had no access to formal instruction or experienced mentors (Maloney et al., 2008). However, further investigation revealed that 111 of the 536 projects contained no scripts at all (they were used for media manipulation and composition) and although it was reported that all of the remaining 435 projects made use of sequential execution; something as simple as snapping two blocks together was counted as sequential execution because it was classified as a stack with more than one block. Interviews also revealed that most of the youth did not identify scripting in Scratch as a form of programming. When asked what computer programming was, they responded that they had no clue (Maloney et al., 2008, p. 370).

Another study found that although student projects showed evidence of computational concepts in code, interviews revealed significant conceptual gaps and students could not explain how their code worked (Brennan & Resnick, 2012). Results

such as these are nothing new. Research conducted over three decades ago investigated 11 and 12 year olds who were able to write and interpret short, simple programs after more than a year of self-guided discovery with the LOGO programming language, but the children had trouble with programs involving fundamental programming concepts and interviews revealed many misconceptions about how the programs worked (Kurland & Pea, 1985). There is growing recognition of the problem of using code to infer understanding of programming concepts and there has been call for research to validate the method of analyzing code because the presence or absence of constructs "may be due to the specific programming environment used or simply students' desire to use the tool in a particular way" (ACM, Code.org, CSTA, Cyber Innovation Center, & National Math and Science Initiative, 2016, p. 206).

There are noted issues with previous studies of Scratch and research suggests that there are other technology tools that may be equally or more effective at helping students learn programming and computational thinking concepts. A literature review of technologies used in Maker philosophy found that Scratch was used the most and the researchers advocated for investigating less common technologies (Papavlasopoulou et al., 2017). Additionally, researchers in another study that chose to use Scratch for their intervention suggested that positive results similar to theirs should be observed when adapting any environment to teach middle school students (Armoni, Meerbaum-Salant, & Ben-Ari, 2015). While Scratch has been commonly used and widely studied, there is need to investigate the potential of other technologies to introduce programming to students. The current study will address this gap by comparing the relative effectiveness

of learning coding skills with Scratch versus an artificially intelligent robot called Cozmo.

Educational Robotics

Robotics has been endorsed as an educational tool by many researchers (Petre & Price, 2004, p. 147). Johnson (2003) argued that robotics are motivating for students because they are concrete and complex and they hold pedagogic value for teachers because making them work allows students to use and extend their knowledge to diagnose and fix problems.

Prior to 2012, no systematic reviews involving robotics in education could be found (Benitti & Barreto, 2012). Benitti pointed out a lack of rigorous quantitative research on educational robotics. Of 70 articles found that discussed the effectiveness of robotics as a teaching tool, only ten employed quantitative methodologies. However, 40% of the studies that used experimental designs included methodological flaws. All but one of the studies used LEGO robotics in the educational activities (the other used a robot developed by the authors) and most of these activities were not integrated into classroom activities; they were done in after-school or summer camp programs. In general, the results of the studies showed learning gains with the use of robotics, but there were cases where there was no significant increase in student learning. Many studies focused on self-directed learning experiences that significantly increased learning in STEM (Science, Technology, Engineering, and Math) areas, but some of the studies reported nonsignificant increases and it was impossible to isolate the variables that

contributed to the success due to methodological challenges (p. 986). Thinking skills, science process skills/problem-solving approaches, and social interaction/teamwork skills were the common focus of the robotics studies, but the results were mixed. Additionally, none of the articles included studies with students aged 11-12. Educational robots seem to be a relevant tool for improving learning, but the assertion needs to be supported with empirical evidence to discover how to use robotics to develop specific skills.

More recent analyses show that the use of robotics in education is expanding, yet some of the same trends continue. A review of the applicability of robots in education found that the two main categories of learning activities were robotics/computer education and nontechnical education (science and language) (Mubin, Stevens, Shahid, Mahmud, & Dong, 2013). A review of the use of robots in education from early childhood through secondary school found that the influence of robots on children's skill development could be grouped into four major categories: cognitive, conceptual, language and social (collaborative) skills (Poh et al., 2016). The pedagogical theories underpinning research on robots in education were also featured and it was suggested that the constructionism paradigm of Papert best fits the field and has been adopted the most in robotics curricula (Mubin et al., 2013, p. 4). Also common was Vygotsky's social constructivism, which applies to peer or tutor-based methodologies and gave rise to the idea of scaffolding (i.e., chunking tasks into smaller bits) (Mubin et al., 2013). The majority of the papers reviewed in a recent study by Poh et al. (2016) were nonexperimental studies and again the majority of the studies used LEGO Mindstorms robots.

As new technology develops and more options in robots become available, an important consideration becomes which robot to use to achieve the set goals. Mubin et al. (2013) highlighted a variety of robotic kits available ranging from low-cost single function kits (e.g., OWI-9910 Weasel) to kits with the option of teaching about robotics and electronics (e.g., Arduino, Parallax BoeBot) to LEGO Mindstorms to humanoid robots (e.g., NAO) costing thousands of dollars. In spite of the increasing options, the use of LEGO Mindstorms still dominates the literature at all levels of education (Cheng, Sun, & Chen, 2018; Karim, Lemaignan, & Mondada, 2016; Mubin et al., 2013; Poh et al., 2016; Spolaôr & Benitti, 2017) and tends to be the most popular choice of educational robot worldwide (e.g., Russia (Ospennikova, Ershov, & Iljin, 2015)).

LEGO Mindstorms appears most often in studies of educational robotics at the early childhood through high school level, but other types of robots are making their way into studies at this level including ROBOLAB, Picocricket, Bee-bots and Pro-bots, Zigbee, Rocky robot, POWERTECH robot, BEAM robot, Parallax robot, GENTORO robot, humanoids, Tangible K, Evobots, Pekee robot, CHERP and Roball (Poh et al., 2016). LEGO Mindstorms robots were used in seven out of 15 studies in the most recent systematic review at the tertiary level of education followed by the use of virtual robots (three studies), and then single studies investigated Boe-bot, observatory, Parallax sumobot, .NET Gadgeteer, and Sphero (Spolaôr & Benitti, 2017).

A review of robots and their impact on K-12 STEM education discussed the three main platforms: brick-based robotic toolkits, modular robotic kits, and pre-assembled robots (Karim et al., 2016). Brick-based kits require children to build the robot first

before it can be programmed. Two such kits include LEGO Mindstorms (Figure 1) and

Jimu (Figure 2). LEGO Mindstorms offers different kits that can be programmed with

drag and drop blocks and Jimu offers different kits that can be programmed with Blockly

coding.

TRACK3R	SPIK3R	R3PTAR
GRIPP3R	EV3RSTORM	BOBB3E
BANNER PRINT3R	RAC3 TRUCK	DINOR3X
KRAZ3	EV3D4	EL3CTRIC GUITAR
EV3MEG	EV3 GAME	MR. B3AM
ROBODOZ3R	WAC K3M	
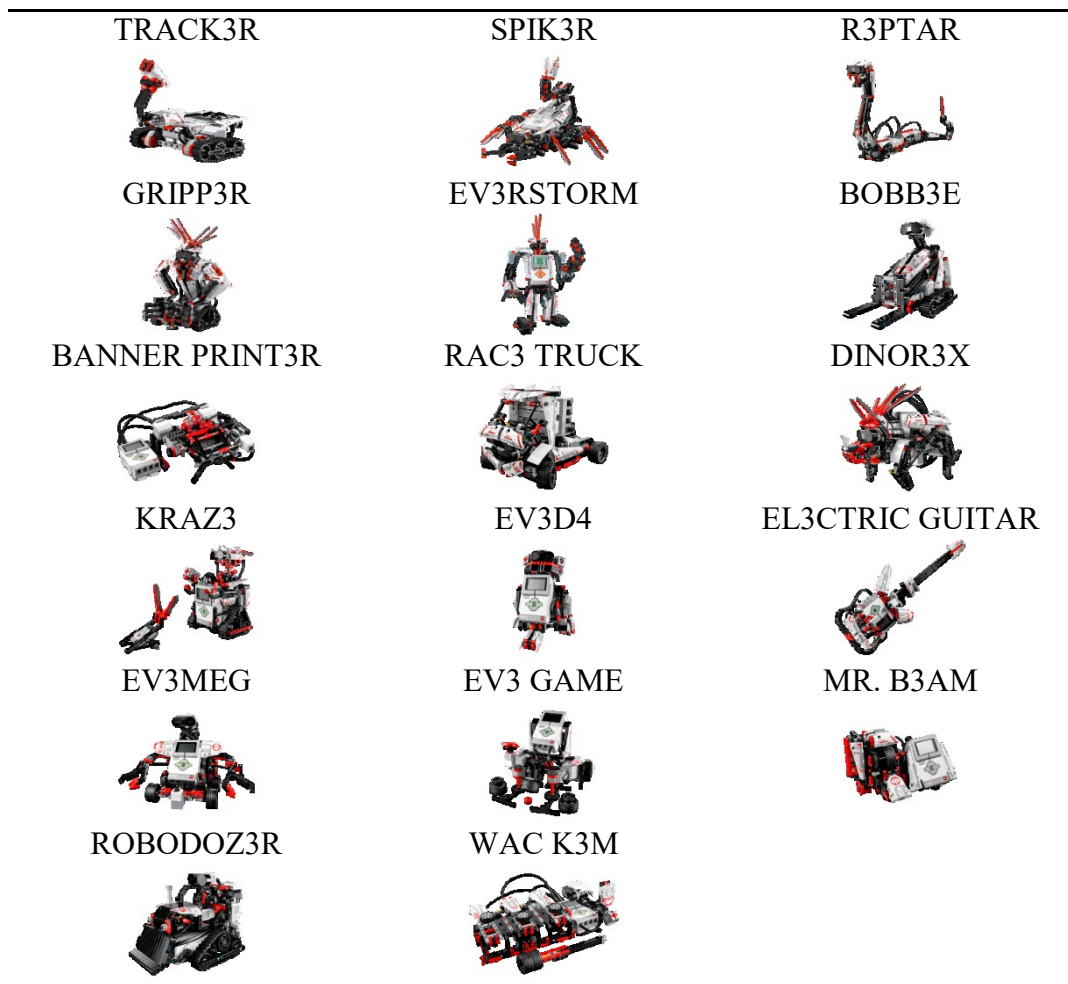
Figure 1. LEGO Mindstorms EV3 robot options. Retrieved from

https://www.lego.com/en-us/mindstorms/build-a-robot.

| BuilderBot | TankBot | AstroBot | MeetBot |
| Buzz/MuttBot | Explorer | Animal Add-On | Inventor |
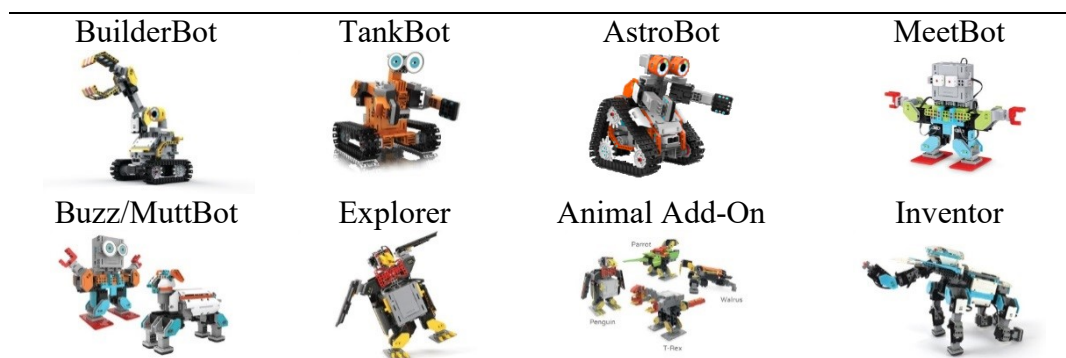
Figure 2. Jimu robot kits. Retrieved from https://ubtrobot.com/collections/jimu-robots.

The brick-based robot kit used the most in educational activities has primarily been LEGO Mindstorms (Cheng et al., 2018; Karim et al., 2016; Mubin et al., 2013; Poh et al., 2016; Spolaôr & Benitti, 2017). Karim et al. (2016) pointed out that even though Mindstorms have been widely used in STEM education; most activities are short-termed and developed informally through extra-curricular activities:

The reason is primarily associated to the time consuming unintuitive overwhelming design process which requires excellent inventory and project management skills. As a direct consequence, teachers' control over the classroom is reduced, which worsens due to the absence of formal structured curricula linking traditional and robot-based education. Thus the role of the teacher as facilitator, educator, or guide is minimized. Most importantly, given the constrained budget in primary and secondary schools, these kits are not always affordable. (Karim et al., 2016, pp. 2-3)

Karim et al. (2016) also discussed modular robotic kits. These kits are not as complex to construct which reduces assembly time, they are more reasonably priced, and

they use limited design space which makes them more convenient for classroom use. However, some of the open source robot designs are not commercially available; they can be made using off-the-shelf components but the design process is very involved (i.e., soldering, wiring, etc.) and the design can be limited.

There are also preassembled robots available, some of which have been marketed as toys to teach children coding. One example is the Ozobot which is programmed with Ozoblockly, a programming language that uses drag and drop blocks to code the robot. Figure 3 shows the two Ozobots that are currently available. The Bit model was developed first, followed by the Evo model (released in 2017) which was designed to be social; Evo makes sounds meant to mimic human emotion. The designer reported that the Bit version is already in about 2,000 schools (Alba, 2016).
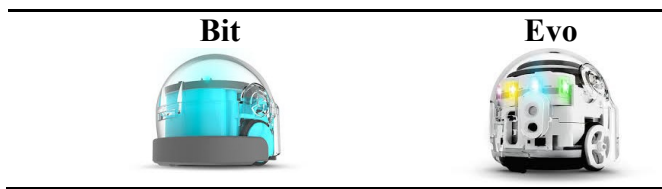
Figure 3. Ozobot models. Retrieved from https://ozobot.com/products.

Ozobots were used to introduce computational thinking ideas to third graders in a map project (Merino-Armero, González-Calero, Cózar-Gutiérrez, & Villena-Taranilla, 2018). The Ozobots improved student motivation compared to a paper-pencil control group, but not student confidence (and males exhibited higher confidence than females). While the study mentioned that student computational thinking and mental rotation were assessed, the results were not reported in the study and the authors suggested that future

studies investigate whether the use of educational robotics improves learning when introducing computational thinking.

The number of preassembled robots with the aim of helping people learn to code is slowly increasing. Another such robot that is marketed as a toy is Cozmo (released in 2016). Cozmo is also a social robot (see Figure 4), but has capabilities beyond those available with Evo and most other robots that are currently available.

Cozmo

Figure 4. Cozmo. Retrieved from https://www.anki.com/en-us/cozmo.

Cozmo is an artificially intelligent robot with computer vision, animation, and emotions that has been marketed by Anki (a consumer robotics and artificial intelligence company founded in 2010 by a trio of Carnegie Mellon graduates with PhDs in robotics) as a toy for ages 8 and up.

Cozmo's software development kit, or SDK, provides access to a variety of sophisticated features through relatively simple lines of code (Gorman & Ackerman, 2017), but to use the SDK requires knowing how to code (Python). In order to overcome this obstacle and make it accessible to non-programmers, the developers designed Code

Lab. Code Lab is part of the Cozmo app that can run on a smartphone or tablet (the app communicates with the robot via Wi-Fi). The Code Lab app adds a graphical user interface (GUI) on top of the SDK using an open-source version of the visual programming language Scratch called Scratch Blocks, which makes using it similar to coding in Scratch. Colorful interactive blocks represent different functions and dragging and dropping these blocks (along with the ability to edit parameters) allows the user to get Cozmo to do all kinds of custom behaviors.

The literature has already demonstrated the potential for Scratch to help students learn programming skills (Brennan et al., 2011). Cozmo is based on Scratch Blocks and also meets Papert's requirements. It has a "low floor" with Code Lab's *Sandbox Mode* (icon-based horizontal grammar) but it also offers a "high ceiling" with the ability to increase in complexity from Code Lab's *Constructor Mode* (text-based vertical grammar) to the Python SDK. Cozmo also offers "wide walls" with the ability to support many different types of projects to appeal to people with different interests.

Cozmo is already showing potential for helping people of all ages learn STEM concepts. Smithsonian Magazine named Cozmo the best overall STEM toy of 2017. Georgia Tech offers an elective course (CS3630 – Introduction to Perception and Robots) for students pursuing the Intelligence thread in the College of Computing's Bachelor of Computer Science program. Professor Chernova chose to use Cozmo for the course because "We were looking for a small, portable robot with a built-in camera sensor and open-source development tools. We ultimately chose to use the Cozmo platform because, in addition to meeting these requirements, the robot is extremely expressive,

with over a hundred built-in animations" (Giles, 2017). Carnegie Mellon University also offers a Cognitive Robotics course that uses Cozmo (C. Hermans, 2017). iD Tech, a company that runs summer STEM education camps for ages 7-18 at Stanford, NYU, and 150 campuses, offered two Cozmo-based courses in 2018 (iD Tech, 2018). Michigan-based company Kinvert integrated Cozmo's Code Lab and SDK to teach K-12 students STEM education. They found that Cozmo is in a league of its own compared to other robots "from personality to student engagement to software potential" and that girls in particular "love using Cozmo when compared to other platforms" (Kaiser, 2018). Kinvert also uses Cozmo at events like birthday parties, Boy/Girl Scouts events, at libraries and public school events.

Virtual robotics (simulated on a computer screen) can fail in the physical world, but the use of an actual robot in contact with its environment can change that. Robotics makes programming tangible by moving the rendered results from a screen to the physical world. While Scratch uses two-dimensional figures on a flat computer screen, Code Lab allows students to watch Cozmo physically act out what they have programmed in the real world. Cozmo can also recognize students' faces, interact with them, and display animations and emotions. Cozmo is already demonstrating potential in a variety of settings (e.g., higher education, STEM camps and workshops), but empirical evidence of its use as a learning tool in K-12 public schools is lacking. Research is needed to investigate the impact that Cozmo may have on programming competency and engagement for students in a public education setting.

Although there is a lack of research evaluating educational benefits of using artificially intelligent robots with facial recognition and emotional support for learning programming skills, there is a growing body of literature on how emotional design affects learning experiences.

Emotions, technology, and learning

Emotional Design developed to promote positive emotions (Norman, 2007) or pleasure in users (Jordan, 2002; Green & Jordan, 2003) by means of design properties. Design based on emotions can affect user experience because emotions affect attention and memory, generate meaning, and influence decision making (Van Gorp & Adams, 2012). There are two main approaches to applied emotional design: one is based on modifying the aesthetic appearance or interface of the object and the other is based on promoting fluent and engaging interactions (Triberti, Chirico, Rocca, & Riva, 2017). Several studies showed the importance of emotional aspects of technologies. Studies in multimedia learning (Um et al., 2012, Plass et al., 2013) showed that embedding emotional stimuli (e.g., face-like shapes, vibrant colors) into interfaces elicited positive emotions in learners and improved learning outcomes. Studies of children's perceptions of different types of technology can provide additional guidance for researchers and educators.

Twenty-six three to ten year old children interacted with Amazon Alexa (a cylinder-shaped customizable, voice-controlled digital assistant with a female voice), Google Home (a cylinder-shaped female voice-controlled device for home automation),

Cozmo (an autonomous toy robot), and Julie Chatbot (a conversational chatbot with text-to-speech voice run on a tablet through an Android app); and then an interactive questionnaire (The Monster Game in which two monsters would share a belief about an agent and then children placed a sticker closer to the monster they most agreed with) was given to gather their perceptions (Druga, Williams, Breazeal, & Resnick, 2017).

Most of the children agreed that the agents were friendly and trustworthy, and they used gender interchangeably when talking about them. For example, some children were not sure whether Cozmo was a boy or a girl and one boy concluded that Cozmo was a bobcat with eyes. The 3-4 year olds very much enjoyed playing with Cozmo. The older children enjoyed interacting with all of the agents, but had their favorites based on the different interaction modalities. The children quickly became fluent in voice interaction and even tried to talk with the agents that did not have this ability (Cozmo). When investigating interactive engagement, the researchers found that although the children were attracted to the voice and expressions of the agents at first, they lost interest when the agent could not understand their questions. Two of the children, Gary and Larry, liked interacting with Cozmo the most "because she could actually move" and because Cozmo had expressions:

> He has feelings, he can do this with his little shaft and he can move his eyes like a person, confused eyes, angry eyes, happy eyes…Everybody else like they didn't have eyes, they didn't have arms, they didn't have a head, it was just like a flat cylinder. (Druga et al., 2017, p. 599)

The authors explained that "Through its eyes and movements, Cozmo was able to effectively communicate emotion, and so the children believed that Cozmo had feelings and intelligence" (p. 599). They concluded that the children believed they could teach and learn from the agents, and they hoped to design interactions where children could tinker with and program agents to expand their perception of their own intelligence and different ways to develop it.

Purpose

This study aimed to examine the effects of the coding activities supported by the artificially intelligent, animated emotional-educational robot Cozmo on middle school students' computational thinking, spatial skills, competency beliefs, and engagement compared to the more traditional computer-based program of Scratch.

Computational thinking skills were measured using the Computational Thinking Test developed by Román-González, Pérez-González, and Jiménez-Fernández (2017). Spatial skills were measured using a revised version of the Mental Rotations Test developed by Vandenburg and Kuse (1978). Competency Beliefs were measured using an adapted version of Activation Lab's Competency Beliefs in STEM survey version 1.0 (2017). Student engagement was measured using an adapted version of Activation Lab's Engagement in Science Learning Activities survey version 3.2 (2016).

Research question

Is there a difference in computational thinking performance, spatial abilities, competency beliefs and engagement among middle school students who are taught coding using a traditional Scratch approach versus students who are taught using the same coding activities with Cozmo, an emotional-educational robot?

Hypotheses

Hypothesis 1: Cozmo is expected to produce greater gains in computational thinking.

Hypothesis 2: Cozmo is expected to produce greater learning gains in spatial skills.

Hypothesis 3: On average, both the Cozmo and Scratch training groups are expected to experience an increase in competency beliefs.

Hypothesis 4: Cozmo is expected to produce greater student engagement.

Definition of Terms

Animation – giving life to an object using computer graphics

Artificial Intelligence (AI) – computer systems that are able to perform tasks that normally require human intelligence (e.g., visual perception, speech recognition, decision-making)

Computational thinking – thought processes involved in forming problems and their solutions in a form that can be carried out by an information-processing agent

Computer vision – a field of Artificial Intelligence and Computer Science that aims to give computers a visual understanding of the world

Debug – identify and remove errors in code

ISTE – International Society for Technology in Education

Maker Movement – builds on a person's ability to be a creator of things, a "Maker"

Making – creating

Scaffolding – chunking tasks into smaller bits

SDK – Software Development Kit

Spatial skills – the capacity to understand, reason, and remember the spatial relations among objects or space

STEM – Science, Technology, Engineering, and Mathematics

Tinkering – making discoveries through problem solving (experimenting)

CHAPTER II

REVIEW OF THE LITERATURE

STEM

The lack of qualified individuals needed in Science, Technology, Engineering, and Mathematics (STEM) careers has prompted several initiatives with the goal of expanding interest and opportunity in these fields (e.g., Computer Science Education Week (http://csedweek.org); AP Computer Science course (https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf)). STEM-related concepts and skills are increasingly appearing in school curriculum standards (ACM et al., 2016; NGSS, 2017; Sandars et al., 2016), and yet the challenge of attracting and keeping students in STEM-related courses remains. In Ohio public universities, more than 40% of students who declare a STEM major leave the field before graduation (J. Price, 2010).

Technology is playing an increasingly significant role in everyday life (Schmidt & Cohen, 2013). Employment in computer occupations is projected to increase by 12.5 percent from 2014 to 2024 (Fayer et al., 2017), but filling these positions with people of diverse backgrounds could prove to be a challenge due in part to a lack of access to computer science courses for under-represented minorities (Margolis et al., 2008) and a significant gender gap in students enrolling in advanced placement computer science courses (CollegeBoard, 2017). It has been found that high school and college students have negative attitudes toward computer science (Carter, 2006; Hewner, 2013) and

22

computer programming competence is considered difficult to develop (Wiedenbeck, 2005). Students can face difficulties comprehending central concepts and composing programs that meet certain expectations, and this can contribute to the failure and dropout rates in introductory computer programming courses (Fesakis & Serafeim, 2009).

This review of literature will focus on the factors cited in current research regarding the teaching and learning of programming. The first section examines computer programming and coding instruction through the lens of constructionism. The second section investigates the tools that have been used to teach programming as well as the interactions between technology, emotions, and learning. The third section discusses the important educational outcomes of computational thinking, spatial skills, competency beliefs, and engagement. The fourth section explores the research question, hypotheses and rationale for each. The significance of the study concludes the chapter.

Constructionism

The creators and innovators of technology have a profound influence on the ways others experience it, and expanding interest and access to programming courses can be a positive step toward expanding diversity in the field. Papert (1980) asserted that programmers' biases influence what is available. Innovations in technology drive the need for innovations in the way concepts and skills are introduced to students in order to engage their interest and motivation so that they can become creators rather than mere consumers of technology. The quick pace of technological change and innovation is shifting the focus from teacher centered to student centered instruction. Jean Piaget

believed that students did not learn by simply absorbing knowledge (i.e., knowledge is

not transmitted to students) but rather through integrating new experiences into existing

knowledge structures (i.e., knowledge is constructed in the student's mind) (Siegler,

1986). This notion developed into his theory of constructivism. Kinesthetic and active

approaches (i.e., learning by doing) to instruction are supported by constructivist

principles. Seymour Papert, a protégé of Piaget, extended the constructivism theory

during the rise of technology. Papert's theory of constructionism suggests that not only

do we learn by doing, but we learn best when engaged in building some type of external

artifact, whether a robot, theory, or story (Papert & Harel, 1991).

Working with real tangible objects is central to Papert's constructionism (Papert

& Harel, 1991). Papert and Harel (1991) clarified the relationship between

constructionism and constructivism: Constructionism "shares constructivism's

connotation of learning as "building knowledge structures" irrespective of the

circumstances of the learning. It then adds the idea that this happens especially

felicitously in a context where the learner is consciously engaged in constructing a public

entity, whether it's a sand castle on the beach or a theory of the universe" (p. 2). Papert

breaks with Piaget by ascribing a larger role to the surrounding culture in providing the

student with materials with which to construct (Cejka, Rogers, & Portsmore, 2006). In

the late 1970s, Papert saw the development of something that he believed would

revolutionize learning – the computer.

Papert (1980) observed the potential of computers in education and noted the

issue of mixing old instructional methodologies with new technologies. He believed that

the dissociated model of learning math (i.e., learning by rote in a decontextualized way) was problematic as was saying that people cannot do math just because they do not get the way it is taught in school; they just need to find a different route to get there.

"Piagetian learning" with its emphasis on natural, spontaneous learning in interaction with the environment was different from the curriculum-driven method of traditional schooling. Piaget's epistemology (theory of knowledge) was not concerned with the validity of knowledge but with its origin and growth. For Piaget, the study of people and what they learn and think are inseparable. Papert believed that Piaget's theory was a good context for learning, but that its stage theory was conservative and reactionary and emphasized what children cannot do. This contrasted with his own beliefs as an epistemologist. The psychological perspective of understanding learning focused on laws that governed the learner instead of what was being learned. Piaget believed that separating the learning process from what is being learned is a mistake. Papert placed greater emphasis on the intellectual structures that *could* develop instead of those that do, and on designing learning environments that resonate with them. He advocated for the use of a transitional object that exists in the environment that can make contact with the ideas in a person's mind.

To get closer to answering the question of "why some learning takes place so early and spontaneously while some is delayed many years or does not happen at all without deliberately imposed formal instruction" requires looking at the "child as builder" (Papert, 1980, p. 7). All builders need materials and Papert varied from Piaget in the role he attributed to the surrounding cultures as a source of these materials:

In many cases where Piaget would explain the slower development of a particular concept by its greater complexity or formality, I see the critical factor as the relative poverty of the culture in those materials that would make the concept simple and concrete. (Papert, 1980, p. 7)

Papert believed that the influence of the materials that the culture provides could play a part in determining the order a child develops different intellectual abilities. He looked to the potential of computers to help people learn and believed that teaching the computer to do something could help people understand the purpose and meaning of what they were doing. Learning to program could transform the process of learning; it could make it more active and self-directed, and could concretize and personalize the formal. During that process, debugging has the potential to move people away from thinking of things as right or wrong toward asking how things can be fixed. Using an analogy of the futility of studying the horse drawn carriage in detail as a means to improve transportation, he criticized the practice in educational psychology of studying existing curriculum and pedagogy instead of imagining the possibilities of something different. His idea was not one of chaos or leaving children to their own devices, but rather supporting them to build their own learning.

"Discovery cannot be a setup; invention cannot be scheduled" (Papert, 1980, p. 115). Papert believed that programming provides the opportunity for an authentic research project with real intellectual collaboration between student and teacher as they pursue a problem until it is completely understood. With programming, there is no expectation for things to work on the first try; the question instead becomes how can

errors be fixed, and fixing them requires understanding of what happened. Studying errors rather than forgetting about them could pay off when debugging programs. "Errors benefit us because they lead us to study what happened, to understand what went wrong, and through understanding, to fix it" (Papert, 1980, p.115).

Papert proposed the use of new technologies, such as computers and robots, to change the nature of learning at school (Julià & Antolí, 2016). He developed the first software designed exclusively for use by children, the LOGO programming language, which consisted of a small mobile robot (a turtle) which moved in response to programmed commands relative to its own position (Julià & Antolí, 2016, p. 187). Papert (1980) explained that there were two turtles; the floor turtle (the robot) and the light turtle (the image on the screen). In the LOGO environment, when a child had a question, the instructor was not to give the child the answer but rather encourage the child to "play Turtle" and act it out. He suggested that this taught the child a method (heuristic procedure) rather than an isolated program. Turtle geometry could develop a mathetic (learning) strategy (i.e., making sense of something in order to learn it) and syntonic learning (i.e., acting things out) could make the learning process concrete. For example, the important thing when giving children a theorem is not for them to memorize it, but for them to use it as a tool with which to think.

How effective has LOGO been at accomplishing the goals of constructionism? Conflicting results are common throughout the literature (Voogt, Fisser, Good, Mishra, & Yadav, 2015). A carefully designed task was created to provide the opportunity for "near" transfer from tasks done with LOGO, but there was no evidence that students in

the treatment group who received LOGO instruction for over a year had better planning skills for a non-programming task than their peers with no LOGO experience (Kurland & Pea, 1985). Yet students who were instructed in debugging skills using LOGO did experience transfer to a non-programming environment (they searched for bugs in their process more effectively than non-LOGO students) and the results still showed up four months after instruction ended (Klahr & Carver, 1988). Empirical studies did not find conclusive evidence that programming experience in LOGO could improve thinking skills (Kurland & Pea, 1985; Kurland, Pea, Clement, & Mawby, 1986; R. Pea, 1983). LOGO was a first attempt at developing software designed exclusively for children, but advancements in technology continue to offer new tools to support the theoretical underpinnings of constructionism.

Computer programming/Coding

Programming and coding have been recognized as one of the important competencies that require students to effectively use computational tools and devices to solve real, complex programs (Chao, 2016, p. 202). Although coding and programming are often used synonymously in the literature (Heintz, Mannila, & Tommy, 2016), some researchers acknowledge that programming includes a coding activity but believe there is a difference in the complexity between the two (Lonati, Malchiodi, Monga, & Morpurgo, 2015). An algorithm is an effective procedure to reach a goal in finite time and a program can be defined as an algorithm written in a programming language (Lonati et al., 2015). "The word **code** suggests a further reduction in the degrees of freedom, a constrained bijection between the procedure one has in mind and its machine

implementation. It seems well suited when one wants to emphasize the technological context of the program" (Lonati et al., 2015, p. 170).

The core of computer programming involves using symbolic commands arranged in an appropriate sequence to create a series of actions in order to instruct a computer's behavior (R. D. Pea & Kurland, 1984). Some argue that programming is too difficult for younger students (Lahtinen, Ala-Mutka, & Järvinen, 2005), but others claim that K-12 students can program computers to perform a variety of useful tasks (Wyeth, 2008). Learning and computer scientists found that many of the reported difficulties in teaching programming have been due to the structure and representation of the tasks in traditional computer science instruction (Ben-Ari, 2001; DiSessa, 2001; Guzdial & Forte, 2005).

Block-based languages have existed since the 1980s, but have found recent adoption as tools for programming education (Aivaloglou & Hermans, 2016). Scratch, Alice, Blockly, and App Inventor are all block-based languages aimed at novice programmers. Several studies have shown that block-based languages are powerful tools for teaching programming (Armoni et al., 2015; T. W. Price & Barnes, 2015). For example, Price and Barnes (2015) found that "block programming interfaces can significantly improve novice performance on some programming activities, specifically through increased time on task and quicker, more frequent achievement of programming goals" (p. 98).

Weese, Feldhausen, and Bean (2016) proposed that the use of a block-based programming language can effectively reduce student cognitive load. Cognitive load

relates to the amount of information that working memory can hold at one time. Sweller (1988) said that since working memory has a limited capacity, activities that do not directly contribute to learning overload it and should be avoided. Visual programming languages can reduce cognitive load and "allow students to focus on the logic and structures involved in programming rather than worrying about the mechanics of writing programs" (Kelleher & Pausch, 2005, p. 131). Programming tools can also make computational thinking practices easier because the outcomes can be viewed in the form of animated objects (Lye & Koh, 2014). Visualization can make computational practices (e.g., testing and debugging) less cognitively demanding (Lye & Koh, 2014).

As technology continued to develop and advance, new programming languages were modeled after aspects of LOGO (Utting, Cooper, Kölling, Maloney, & Resnick, 2010). Papert designed LOGO in 1968; work on a robotics versions of LOGO was taking place at the MIT Media Lab in the 1980s; and work on programmable bricks began in the early 1990s (LEGO RCX and NXT grew out of this work) (Lye & Koh, 2014). In 2006, a new version of LOGO called Scratch was developed by the Lifelong Kindergarten Group at the MIT Media Lab (Lye & Koh, 2014).

Tools

Three categories of activities have been proposed for introducing Computer Science to children: kinesthetic (e.g., unplugged activities that do not use technology), visual programming environments (e.g., Scratch), and robotics (Armoni et al., 2015; Levy & Ben-Ari, 2015). Levy and Ben-Ari (2015) argued that compared with kinesthetic

and visual programming environments, robotics has the additional advantage of enabling student to learn a variety of STEM subjects and not just computing. "Robotics are appropriate for introducing CS (and other subjects such as physics and mathematics) to young people because the algorithms and programs are reified in concrete objects and not just as virtual characters on screen, as in Scratch and Alice" (Armoni et al., 2015, p. 25:2).

Scratch

Scratch uses a collection of graphical programming blocks that can be snapped together to create programs. The blocks are shaped to fit together only in ways that make syntactic sense so that novices do not have to learn the complex syntax of punctuation required by traditional programming languages. The designers of Scratch established three core design principles: make it more tinkerable, more meaningful, and more social than other programming environments (Resnick et al., 2009). The name "Scratch" was selected to highlight the idea of tinkering and comes from hip-hop disc jockeys who tinker with music by mixing clips together in creative ways. Resnick et al. (2009) suggested that Scratch is similar because it enables users to mix graphics, animations, photos, music, and sound. Scratch focuses on two-dimensional rather than three-dimensional images based on the belief that it is easier for people to create, import, and personalize 2D artwork (Resnick et al., 2009).

Scratch grammar is based on a collection of graphical programming blocks that can be snapped together to create programs and the scripting area (on the right side of the

screen) in the interface is intended to be used like a physical desktop where users can drag and tinker with blocks (see Figure 5).

Figure 5. Scratch scripting area.

The Scratch blocks are shaped to fit together only in ways that make syntactic sense. Control structures (like *repeat*) are C-shaped to suggest that blocks should be placed inside them (see Figure 6). Blocks for output values are shaped according to the types of values they return; ovals for numbers and hexagons for Booleans (see Figure 7). Conditional blocks (like *if*) have hexagon-shaped voids to indicate that a Boolean is required (see Figure 8).
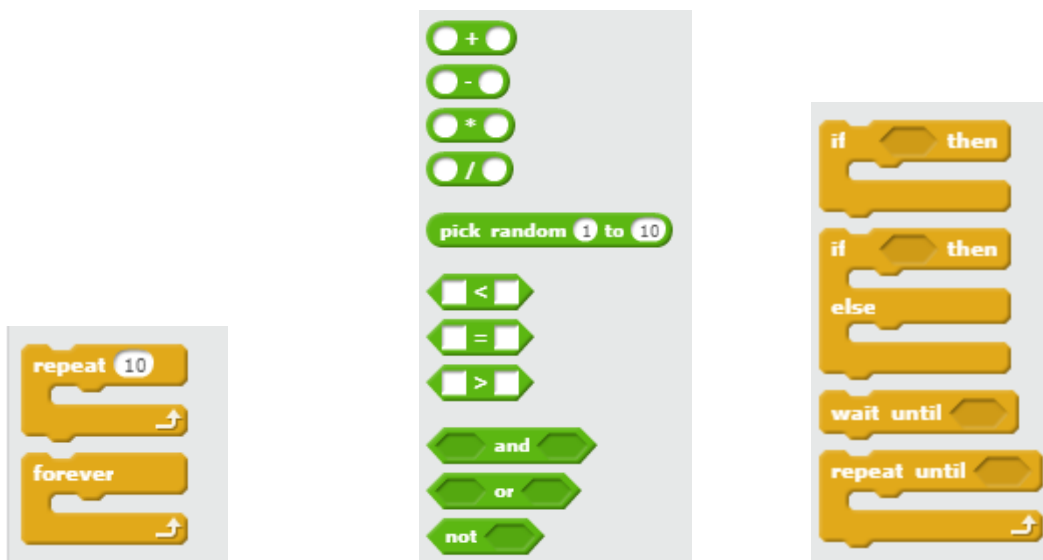
Figure 6. Control structures.   *Figure 7.* Operator blocks.   *Figure 8.* Conditionals.

Mixed results have been reported for the success of Scratch at accomplishing study objectives. Programming in Scratch did not cause any significant differences in the problem solving skills of primary school students even though there was a slight improvement in self-confidence for problem-solving ability (Kalelioglu, 2015). A study comparing Scratch and LOGO for differences in attitude and learning outcomes among a group of academically advanced sixth grade students that applied to participate in the enrichment program found that Scratch helped students understand conditional statements better (it was noted that code blocks in Scratch only lock together in syntactically valid ways so errors are always semantic and never a result of typing or syntax errors) while LOGO boosted confidence in programming ability more (Lewis, 2010). Students that used Scratch in a STEM summer outreach program showed statistically significant positive gains from pre-survey to post-survey in self-efficacy for

computational thinking concepts but not 21st century learning skills (J.L. Weese et al., 2016).

Some studies have aimed to investigate whether learning Scratch first can ease the transition to learning text-based programming languages. One such study looked at the transition from studying Computer Science in middle school using Scratch to studying computer science in secondary school using a professional programming language (C# or Java) (Armoni et al., 2015). It was reported that the students who had learned Scratch first experienced the benefits of needing less time to learn new topics, fewer learning difficulties, and higher cognitive levels of understanding for most concepts, but at the end of the teaching process there were no significant differences in achievement compared to students who had not studied Scratch. Interviews with teachers and students revealed that students recognized concepts from Scratch when they were introduced in the tenth grade course but with restrictions; they recognized them only in the form that they had encountered them in Scratch. Many of the differences in the grades between the experimental group and the control group were not significant. The concept of repeated execution (loops) and the level of relational creating were the two areas that showed significant differences in favor of the experimental group. The authors suggested that the variable nature of the quantitative results "points to a fertile field for future research" (Armoni et al., 2015, p. 2:13).

A study with a similar goal at the tertiary level took a look at the summer version of a Computer Science course at Harvard College. The researchers implemented Scratch in the course with a goal of improving first-time programmers' experiences rather than

improving scores. A survey of students on how their initial experience with Scratch affected their subsequent experience with Java revealed that most of the students without prior background experience felt that Scratch was a positive influence, but 16% of students (who had prior programming experience) felt that Scratch had no influence at all, and an additional 8% felt that Scratch was a negative influence.

Previous works have attempted to analyze static Scratch programs to evaluate various programming concepts (Brennan & Resnick, 2012; Fields et al., 2017; F. Hermans & Aivaloglou, 2016; Maloney et al., 2008). Maloney et al. (2008) claimed that students at an after school clubhouse learned key programming concepts in the absence of instructional interventions or experienced mentors, but closer investigation revealed issues including 111 out of 536 project containing no scripts at all, counting two blocks snapped together as sequencing, and most students failing to identify scripting in Scratch as programming. A more recent study of archived youth Scratch programs showed some improvement over time, indicated broad concepts that users struggled with, and that learning programming by choice did not guarantee introduction to a wider range of computational concepts (Fields et al., 2017). The largest group of projects used a minimal amount of loops (from one to three) and almost no other programming concepts. These users created relatively small and simple projects with few if any advanced kinds of commands. Fields et al. (2017) acknowledged that while frequency counts of blocks have been used before, the approach only reveals profiles on a surface level and is limited as a stand-in for learning because there is no way to see approach to programming and whether blocks are used correctly or functionally.

The results of a controlled experiment conducted by Hermans and Aivaloglou (2016) found that long scripts and code duplication decreased a novice programmer's ability to understand and modify Scratch programs. An analysis of 250,000 Scratch projects in the public repository revealed that most Scratch programs are small (however Scratch programs consisting of over 100 sprites exist), programming abstraction concepts like procedures are not commonly used (the most commonly used blocks are from the Control and Data categories), and Scratch programs suffer from code smells including large scripts (30%) and unmatched broadcast signals (Aivaloglou & Hermans, 2016). More than one quarter of the projects contained dead scripts (Scratch does not indicate that scripts are dead), 11% of the projects used exactly identical clones between sprites (Scratch does not support procedure calls between sprites, only within them), there were 1,700 scripts used in multiple projects (sometimes as often as in 1,600 different projects), and even though 77% of the projects contained loops, only 14% contained conditional loops. The authors attributed the increased use of the *forever* loop to the Scratch language design and were skeptical about whether it indicated an understanding of loops. Automated quality assessment tools have been proposed for identifying bad programming practices in Scratch (Boe et al., 2013; Moreno-León & Robles, 2014).

Concern over how to assess student progress in programming has been expressed in a number of studies (Brennan & Resnick, 2012; G. Chen et al., 2017; Djambong & Freiman, 2016; Shuchi Grover, Cooper, & Pea, 2014; Román-González, Pérez-González, & Jiménez-Fernández, 2017). This worry spills from academic research into the classroom with one teacher interview revealing that "There is little guidance on

assessment and I fear that many schools will just go down a 'death by scratch' approach with some children simply following instructions" (Sentance & Csizmadia, 2017, p. 481).

In addition to the problem of assessment and analyzing code, there are also methodological issues with studies of Scratch. Several studies reported the benefits of Scratch but were based only on self-reported surveys and questionnaires (Malan & Leitner, 2007; Papavlasopoulou et al., 2017). Many of the studies of Scratch displayed a gender imbalance, lack of a control group, self-selection bias (conducted in elective courses or voluntary after-school programs where there is generally higher interest and motivation), learning gains limited to the context where they were studied, and the use of assessments that need to go through a thorough validation process (e.g., Grover, Pea, & Cooper, 2015; Lewis, 2010; Lye & Koh, 2014).

There has also been evidence reported that Scratch falls short when compared to other technologies. In a study comparing Scratch with App Inventor for Android (AIA), self-efficacy (confidence in the ability to perform well in class) increased for both groups, but the increase in intrinsic goal orientation and task value beliefs (perception about the material of the course in terms of interest, importance, usefulness) was higher for the AIA group (Nikou & Economides, 2014). Another study used Scratch and App Inventor for teaching introductory programming in secondary education (Papadakis, Kalogiannakis, Zaranis, & Orfanakis, 2016). The Computer Attitude Scale (Loyd & Gressard, 1984) was used to evaluate how the programming environment shaped the attitudes, perceptions and beliefs of students in programming and the questionnaire programming knowledge (QPK) (Kleinschmager & Hanenberg, 2011) was used to evaluate student programming

knowledge. The findings showed the greatest improvement in the App Inventor group followed by the Scratch group, and the worst improvement in the control group. The authors suggested that these results confirmed the superiority of App Inventor for teaching programming to novice programmers. Each of these studies compared two different virtual programming environments; however, the crux of Papert's theory of constructionism advocates for tools that can simplify and concretize complex concepts and the research is increasingly showing the potential of robots as powerful tools for introducing programming to novices.

Robotics

Robots can be viewed as "specialized computers with both computational and mechanical facilities to perform physical movement-oriented tasks" (Štuikys, 2015, p. 266). Research supports the use of educational robotics in the teaching and learning of STEM concept areas (Barker, Grandgenett, Nugent, & Adamchuk, 2010; Benitti & Barreto, 2012; Rogers & Portsmore, 2004; Williams, Ma, Prejean, Ford, & Lai, 2007). There have been proposals to introduce educational robotics at all educational levels from early childhood (Sáez-López, Román-González, & Vázquez-Cano, 2016) to tertiary (Spolaôr & Benitti, 2017).

The use of robotics to teach STEM is effective because it enables real-world application of concepts while helping to remove the abstractness of science and mathematics (Nugent, Barker, Grandgenett, & Adamchuk, 2010). A robot as a tool "can help make abstract ideas more concrete, as the child can directly view the impact of his or

her programming commands on the robot's actions" (Sullivan, Kazakoff, & Bers, 2013, p. 205).

Robots can be motivating and engaging for learners and are becoming an effective tool to enhance student motivation and learning performance (Chang, Lee, Chao, Wang, & Chen, 2010; Klassner & Anderson, 2003; Mitnik, Nussbaum, & Recabarren, 2009). "Unlike many apps and educational software developed for children, robotics activities do not involve sitting alone, in front of a computer" (Sullivan & Bers, 2016, p. 3). Studies have shown that physical robots could elicit users' social behavior (Astrid, Krämer, Gratch, & Kang, 2010), were perceived as more engaging, enjoyable, sociable, and informative (Fasola & Mataric, 2013; Paauwe, Hoorn, Konijn, & Keyson, 2015), and could lead to better user performance (Li, 2015). A meta-analysis of studies showed that, in general, the use of educational robotics increased student learning of specific STEM concepts (Benitti & Barreto, 2012). Robotics was found to enhance STEM knowledge of elementary and high school students (Nugent et al., 2010) and to positively influence critical STEM abilities including spatial ability (Coxon, 2012) and sequencing (Kazakoff & Bers, 2012; Sullivan & Bers, 2016; Sullivan et al., 2013). Sullivan and Bers (2016) asserted that robotics and programming curriculum have the potential to foster computational thinking in young children. In spite of these positive effects, it is rare to see robotics integrated into K-12 classrooms (Williams et al., 2007).

Several studies have investigated the applicability of robotics in education (Benitti & Barreto, 2012; Cheng et al., 2018; Mubin et al., 2013; Poh et al., 2016; Spolaôr & Benitti, 2017). In a survey of the essential applications of educational robots, Cheng et

al. (2018) found that educational use of robots was the most prevalent (for 5 out of the 6 groups surveyed), but the goals of each age group were different. For preschool, the educational goal of robotics was to enhance and cultivate interest in robotics while for other groups the goal was to support developing skills concerning robotics (e.g., programming, robot-interaction, AI design). Poh et al. (2016) reviewed studies regarding the use of robots in the education of young children, and results showed that robots were able to enhance development of problem solving ability, team skills, achievement scores, science concepts, sequencing and language skills, and participation. Mubin, Stevens, Shahid, Mahmud, and Dong (2013) also reviewed the applicability of robots in education. They found that robotics can be used to facilitate student learning and improve educational performance and they are an improvement on purely software-based learning because they provide embodiment and the ability to add social interaction to the learning context. Robots were primarily used to provide language, science or technology education, and a robot could take the role of tutor, tool, or peer in the learning activity. The pedagogical theories underpinning robotics in education gradually shifted from constructivism to constructionism. Robots provide a tangible and physical representation of learning outcomes and bring added value to the classroom in the form of a stimulating, engaging, and instructive teaching agent.

Gaudiello and Zibetti (2016) made a distinction between *learning robotics* (students use a robot as a platform to learn robotics/engineering), *learning with robotics* (robots are used as assistants for teachers or companions for students), and *learning by robotics* (students learn both about the content of the lesson and about robots by

acquiring subject-specific knowledge and foster four dimensions of learning – cognitive, affective, social and metacognitive).

Robots have been proposed as teaching assistants for children (*learning with robotics*), but there is a lack of empirical evidence for the effectiveness of doing so (Salvini, Korsah, & Nourbakhsh, 2016). One study using the robot as teaching tool scenario used a robotic tutor (NAO T14 from Softbank) and an interactive touch table for the educational activities with children in grades four through six (between the ages of 10-12). The results revealed problems with student comprehension and sense of agency and control. There were several issues noted. The children were not prone to actively listen to the long instructions given in the robot's monotone voice, but also had trouble actually hearing what the robot said because its noisy motors were obstructing its speech. The robot was unable to engage the children initially because it either explained something poorly or failed to explain it at all. The robot was also unable to identify children's misunderstandings; it only perceived that the child answered incorrectly but not *why* which is a central competence for a skilled teacher.

Mixed results can be found in studies of different age groups that compare human and robot teachers. One such study investigated teacher as robot (i.e., telepresence: the teacher is in another location and talks to the class remotely through a screen on wheels) and robot as teacher (i.e., an autonomous robot teaches the class) (Edwards, Edwards, Spence, Harris, & Gambino, 2016). Both lectures were pre-recorded but appeared to be taking place in real-time. The undergraduates in the class perceived both as credible, but the teacher as robot was perceived as more credible and students reported better attitudes

because of enhanced perceptions of source credibility. The robot as teacher had higher behavioral influence (students were more likely to follow behavioral suggestions of the robot). Similarly, 210 children at a summer camp ages 6-16 were divided into three age groups, and then each group was split in two; one group received a lecture from a robot (Baxter – an industrial robot with an animated screen to express itself through facial expressions) and the other group received a lecture by a human teacher (Fernández-Llamas, Conde, Rodríguez-Lera, Rodríguez-Sedano, & García, 2018). Younger students received the lecture from Baxter and older students received the lecture from the human teacher. The only group that did not improve with respect to younger students' scores was the oldest group taught by the human teacher. The researchers observed both conditions and thought that the older students were not paying enough attention in the human-taught class. They reasoned that it could be because it was a summer code camp where students were not expecting to be evaluated and so did not pay much attention to "just another human teacher" while the novelty of Baxter with the other group resulted in students paying more attention. In a study of culturally variable preferences for robot design and use, people had low evaluations of robots as companions or teachers independent of cultural origin (South Korea, Turkey, and USA) (Lee & Sabanović, 2014).

Many studies included an element of *learning robotics* by having students design and build robots as part of the intervention. In their review of robotics in K-12 STEM education, Karim et al. (2016) found that LEGO Mindstorms was the most commonly used in robot educational activities and that most of the activities were short-termed and

developed informally outside of school. They attributed this to several factors including reduction in teacher control of the classroom; absence of formal structured curricula to link traditional and robot-based education; unintuitive, overwhelming, time-consuming design process, and expense of the kits. Not only can cost be a barrier to implementation of robotics in education (Vandevelde, Wyffels, Ciocci, Vanderborght, & Saldien, 2016), but the commercial kits limit the maximum potential of robots (Vandevelde et al., 2016). For example, Vandevelde et al. (2016) noted issues with the LEGO Mindstorms kit including the fact that the programmable brick can only drive three motors; the kit offers limited components; and interfacing with third party components is difficult.

Additionally, it has been found that "the LEGO engineering curriculum brings with it a number of issues in the classroom, making life harder for the teacher" (Rogers & Portsmore, 2004, p. 23). Rogers and Portsmore (2004) list among potential issues the following: Batteries can die; computers can crash; students can mistakenly program their controller; teaching style and material is different from conventional methods; hands-on projects have no single right answer and teachers may not know the answers to student questions; discussing and answering student questions is difficult when the teacher has to change batteries, reboot computers and get kids to share the LEGO bricks; and teachers lack training and need support in the classroom.

Experimental methods are lacking and quantitative analysis of robotics in education is needed (Benitti & Barreto, 2012; Poh et al., 2016). In an effort to systematically review quantitatively assessed robot applications grounded in learning theories, Spolaôr and Benitti (2017) found that only 15 out of 1,416 studies qualified. Of

these 15 papers, 7 used LEGO Mindstorms robots. The concepts taught in the studies were concentrated in two areas: learning of concepts/subjects and skill development. Robots were used as a tool to support computer science concepts (especially programming) in 60% of the studies. Engineering and physics, psychology, and astronomy concepts were also addressed and many of the studies were interdisciplinary. Teamwork, communication and problem solving were the most commonly addressed skills. Innovation, self-concept and creativity were also studied. Spolaôr and Benitti (2017) concluded that using robotics as a teaching tool, grounded in learning theories, can support learning subjects not directly related to robotics.

A systematic literature review conducted by D.-J. Liu, Huang, Chen, and Fan (2016) revealed that in most cases, the robot plays the role of a teaching tool (i.e., *learning by robotics*) as opposed to playing the role of a teacher or learning companion (i.e., *learning with robotics*).

> There are important differences between using a computer, which runs a model and simulates the results on a screen, and a robot running the same model by manipulating physical equipment. Firstly, the visualization on the screen is two-dimensional, while with the robot, the environment is three-dimensional. Secondly, in computer simulation interaction with the user is limited and non-natural, whereas with the robot, when manipulating objects physically, the interaction is more natural and the results are more understandable to students. (Fernández-Llamas et al., 2018, p. 462)

Gaudiello and Zibetti (2016) echoed support for learning by robotics but pointed out that it takes more than a robot to improve outcomes:

Learning by robotics enhances mathematics knowledge and competencies, has a strong impact on the affective, social, cognitive and meta-cognitive dimensions of learning, and triggers a profound transformation of students' attitudes toward learning and teachers' attitudes toward teaching. However, these effects are not the results of robotic-based activities alone, but of the scaffolding by an appropriated pedagogical approach. (Gaudiello & Zibetti, 2016, p. 144)

The constructionist learning approach is advantageous because it can increase motivation and depth of understanding of the subject at hand (Papert, 1980; Stager, 2005) and it is gaining increased attention in the STEM fields (Fortus, Krajcik, Dershimer, Marx, & Mamlok-Naaman, 2005; McPherson, 2014). Many studies in programming (K-12 and higher education) have reported positive exposure to a constructionism-based learning environment with an authentic problem, scaffolding, and reflection activities (Lye & Koh, 2014).

Most of the literature on robotics in education has been exploratory, anecdotal, and descriptive in nature, and there is a lack of studies that have used qualitative or quantitative methods to explore robotics activities (Williams et al., 2007). Although many qualitative data show that educational robotics can help students learn subject related knowledge (content knowledge) and academic skills, evidence backed up with quantitative data is rare (Eguchi, 2009).

Studies investigating *learning by robotics* have found mixed results with respect to the cognitive and affective effects of using educational robots. Some educational robotic studies have emphasized the duration of involvement; in general longer robotic interventions (e.g., forty hours or more) affected content learning while short interventions (e.g., three hours) affected motivation and attitude (Karim et al., 2016; Nugent et al., 2010) but not all longer interventions (e.g., one year) improve performance (Hussain, Lindh, & Shukur, 2006; Lindh & Holgersson, 2007).

Robotics have been used in an effort to improve cognitive skills (Caci, Chiazzese, & D'Amico, 2013; Caci, D'Amico, & Chiazzese, 2012) and several studies highlight positive results for robotics interventions. For instance, children enjoyed robotics programming and results showed that performance on geometric thinking and metacognitive tasks were improved when programming (Keren & Fridin, 2014). Kazakoff and Bers (2012) used CHERP tangible programming blocks or interface to program robots at two schools (one public, one private) in Massachusetts with one control and one experimental group at each school. (The control groups did not participate in comparable activities in either school – they completed art activities and the regular curriculum). The experimental groups increased their scores in sequencing skills but the control groups did worse from pretest to post-test.

Preliminary findings reported on a study in Austria and Sweden of 148 students (mean age 14.9 years) who completed a pretest questionnaire, eight months of robotics activities, and a post-test questionnaire revealed that the experimental group scored higher than the control group on the pre- and post-test questionnaire (Kandlhofer &

Steinbauer, 2016). In the experimental group, 85% of students attended weekly activities at school to prepare for the national RoboCup Junior (RCJ) competition and 15% of students attended weekly robotics electives at school using the LEGO Mindstorms NEXT platform while the control group attended comparable subjects and activities in computer science elective courses or other electives in media, physics, chemistry or the arts. Both experimental groups showed significant improvements in several subscales: there was a significant positive impact of robotics on math and science investigations, teamwork, and social skills. There was also a significant positive impact of robotics on technical skills and social aspects/soft skills. Significant positive relations were also found between various subscales: computer science and textual programming; general programming/robotics and math/science investigation, adoption/enjoyment of science, robotics self-efficacy; problem solving, teamwork; and attitudes and soft skills.

Özüorçun and Bicen (2017) observed that scores in an engineering course (algorithm writing) were low. Students were running their code on computers, and the researchers wanted to see if robots would improve learning/scores. They implemented a robotics intervention and found that the understanding of the group using robotics improved significantly from pretest to post-test while the control group that did not use robotics did not. It was also reported that students in the robotics group had a positive attitude and learned the programming algorithms well.

In addition to cognitive skills, studies of robotics have also investigated their impact on affective domains. Levy and Ben-Ari (2015) investigated one population of middle school students participating in the FIRST LEGO LEAGUE Competition

(extracurricular and self-selected) and a second population of middle students participating in a new program of the Ministry of Education (part of the school curriculum with students selected by teachers and principals). Analysis of answers from 106 questionnaires showed attitudes and subjective norms were not as high as expected, but scores for student intentions to continue studying STEM were very high, which implies that students are likely to choose to study STEM in the future. "Robotics activities are often justified on the claim that they motivate students to pursue further studies in STEM subjects. Our results provide empirical evidence that supports this claim" (Levy & Ben-Ari, 2015, p. 29).

While positive results are encouraging, not all robotics interventions achieve their intended aims. For example, knowledge gain between the experimental group and control group in a middle school classroom using LEGO robotics was not different overall, but average performers in the experimental group showed statistically significant higher gain in mathematics knowledge after the year-long curriculum exposure (Lindh & Holgersson, 2007). A summer robotics camp (two weeks, two and a half hours per day) for middle school students used LEGO Mindstorms robotics kits and the RoboLab programming environment to complete challenges in small and whole group activities. Results showed that physics content knowledge improved but not scientific inquiry skills (Williams et al., 2007).

Other robotics studies also found minimal gains in student learning of science concepts. For example, Nugent et al. (2010) found that students at a summer camp displayed significant gains in test scores for the content areas of mathematics, computer

programming, and engineering/robotics, but there was no significant improvement in scores related to science learning of geospatial concepts. Two factors offered as explanations were the lack of alignment of the geospatial concepts taught through the robotics activity and the middle school science curriculum, and the short length of time of the intervention. While students participating in the three hour intervention did not demonstrate increases in their understanding of STEM concepts, the shorter intervention did have a positive effect on student attitudes and interest in STEM.

Additional studies focused on student attitudes show mixed results. Markham and King (2010) taught a computer science course (CS1) with one section using the Scribbler robot and a second group serving as a control. Using surveys, they found that the robotics group devoted more effort compared with the non-robotics class. However, McGill (2012) conducted a similar study with students enrolled in a computer science course (CS0) who did not intend to specialize in computer science and found that the Scribbler robots improved students' attitudes towards programming, but had little effect on other measures such as confidence.

Robotics has been one of the tools used to develop computational thinking in students, but it is rare to see robotics integrated into K-12 classrooms (Williams et al., 2007). Efforts are being made to educate preservice teachers on how to develop computational thinking skills with educational robotics with reports of increased teacher computational thinking (Jaipal-Jamani & Angeli, 2017), but the educational robotics research completed since Benitti's 2012 analysis reveals that some of the same trends continue.

Fifth through eighth grade students from rural Wyoming (including a school from an American Indian reservation) participated in one of two methods for a gaming intervention using LEGO EV2 robotics kits and MINDSTORMS software to program them in the robotics context, and Scalable Game Design software including AgentSheets (two-dimensional) or AgentCubes (three-dimensional) in the gaming context (Leonard et al., 2016). Two different methods were used for the gaming intervention: tutorial first then project completion versus project first where students came up with an idea for a game and then learned to code without a tutorial. The project first method had better results than the tutorial first method. Attrition and incomplete surveys made the data set small for this preliminary study. Pre- to post- self-efficacy scores on the construct of computer use declined significantly while the constructs of video gaming and computer gaming remained unchanged. When analyzed by type of learning environment, self-efficacy on video gaming increased significantly in the combined robotics/gaming environment compared with the gaming-only context. Student attitudes toward STEM did not change significantly as a result of the study and computational thinking strategies varied by method of instruction: students who participated in holistic game development had higher computational thinking ratings (Leonard et al., 2016).

A three year project on computational thinking funded by the National Science Foundation with the same lead author involved training teachers in grades four through six to implement Scalable Game Design and LEGO EV3 robotics during after school clubs (Leonard et al., 2017). Thirty teachers and 531 students took part in the study that blended game design and robotics. Eight teachers and 98 students participated in a large

urban city in Pennsylvania while the remaining 22 teachers and 433 students participated in rural Wyoming. Quantitative data were all survey-based and qualitative data included field notes/ratings. The pre-survey ratings revealed that teachers went from *almost never* and *sometimes* integrating STEM content and applied knowledge into their instruction to *sometimes* doing so on the post-survey. Researchers observed teacher practices and rated them on success with student learning and the learning environment. These ratings were slightly higher on robotics versus game design at baseline. The robotics scores dropped in the final observations, but there were no details reported to suggest how or why. The ratings of teacher practices in game design increased, but the lowest rating was on the "Relevance" dimension (Leonard et al., 2017).

With the growing mix of graphical (i.e. onscreen), tangible (physical, hands-on), and hybrid (combined graphical and tangible) interfaces that are available to teach computational thinking skills, careful choices about the learning affordances of interface types must be made (Pugnali & Sullivan, 2017). Studies have demonstrated the use of robots compared to simulators (Mitnik, Recabarren, Nussbaum, & Soto, 2009). Some studies found that students failed to give appropriate directional commands to characters in virtual environments (e.g., code.org (Kalelioglu, 2015) and Java applets from the National Library of Virtual Manipulatives (Fesakis et al., 2013)). Robots can make it easier to find and fix errors in simulations sooner because most of the time the robots provide direct evidence of solution accuracy (Fronza et al., 2017, p. 113). Some researchers reported no significant differences between learning gains for students who participated in a virtual or physical version of the curriculum (A. S. Liu, Schunn, Flot, &

Shoop, 2013) while others have suggested that the embodied nature of physical robots may provide learning benefits for novice programmers (Silk, Schunn, & Shoop, 2009). Auerbach, Concordel, Kornatowski, and Floreano (2018) highlighted three open-ended student projects that demonstrated the ways that students learned from discrepancies between simulation and reality using the RoboGen platform.

A study that investigated the impact of user interface on young children's (ages four through seven) computational thinking (Pugnali & Sullivan, 2017) compared the use of ScratchJr on the iPad with the use of the tangible robotics kit KIBO (developed by the authors). With KIBO, children assemble their own mobile robot with motors, wheels, and sensors and program it to move by connecting interlocking wooden programming blocks without the need for any screen time from tablets or computers. The study reported high mastery of sequencing, repeat loops, and conditional statements for both groups and the lowest scores in debugging for both groups (it was mentioned that debugging was not explicitly taught in the curriculum). The KIBO group performed better on average on every Solve-It task assigned, and they mastered sequencing and debugging significantly better than the ScratchJr group.

It is important to note that the study took place in a summer program that required a $200 registration fee (with no scholarships offered for those who could not afford it) that accepted participants on a first-come, first-served basis. The KIBO sessions reached capacity a month before the start date so there were participants who wanted to be in that group but were unable. None of the ScratchJr sessions reached capacity. This resulted in there being a significantly higher number of boys ($n = 10$) than girls ($n = 2$) in the KIBO

group and a higher number of girls ($n = 8$) than boys ($n = 6$) in the ScratchJr group. Pairs of students shared one robotics kit during the program while each child in the ScratchJr group had access to his or her own iPad and could easily work individually. Additionally, the KIBO robot provides feedback when a child creates a syntactically incorrect program (it makes a noise indicating something went wrong and does not perform any actions) while characters in ScratchJr will execute any programming blocks given and it is up to the child to realize that what was programmed does not match the actions on the screen. The authors expressed intent to study different tangible and graphical technologies other than KIBO and ScratchJr to provide insight about whether their results were specific to affordances of the particular interface or simply to the two different technologies themselves.

Another study comparing a virtual (Scratch) to a tangible (LEGO EV3 Robotics kit) environment was conducted with sixth and ninth graders in a school in New-Brunswick, Canada (Djambong & Freiman, 2016). In this pilot study, students took a pretest, completed a 5 week curricular unit with their assigned program (sixth graders – Scratch; ninth graders – LEGO robotics), and then took a post-test (paper/pencil, multiple choice) using tasks from the Bebras international challenge on informatics and computational thinking (Dagiene & Stupurien, 2015; Dolgopolovas, Jevsikova, Dagiene, & Savulioniene, 2016). Average scores increased slightly from pre- to post-test for both groups but a breakdown of scores by task difficulty and computational thinking skill involved showed decreases in scores in several areas. The researchers concluded that each of the following may have contributed to student ability to solve tasks: type of

programming environment, complexity and structure of problem solving activities (more difficult tasks resulting in lower scores in general), and computational thinking competence involved in a task. There was no clear relationship between task competence composition and average score earned on the assessment which led the authors to believe that the instrument they used for assessment did not appear to be valid.

There were several limitations for the study including small sample size (lacks generalizability), lack of random assignment, paper/pencil assessment (could be demotivating since students were used to using computers), multiple choice questions (students can randomly guess and it is not possible to tell their thought process), and the study did not mention what the five curricular units were. Different programs were used with different grade levels and teachers and it was unclear what activities were used with each group. While it was noted that it was a pilot study, the lack of information makes it difficult to draw any firm conclusions or understand what contributed to the reported scores.

While some studies focus on comparing tangible to graphical interfaces, others look more closely at the nature of the programming languages themselves. One of the few studies that took place in a public elementary school involved the authors developing their own multiple choice and open-ended question assessment of computational thinking (G. Chen et al., 2017). They prepared two forms of predefined programming languages (*text-based* similar to most professional programming languages, and *visual* similar to Scratch that uses drag and drop programming) but kept the problems identical. They used the assessment as a pretest and post-test with two different fifth grade classes (one

high and one low) after teaching a robotics curriculum in which students wrote programs, tested them on a virtual robot, then took turns running them on one physical robot. The robotics course used a visual programming environment and one of the goals was to examine whether students would respond differently to text-based or visual items on the assessment.

In the lowest class, student performance improvement was significant and effect size moderate. In the highest class, student performance improvement was significant and effect size was large. In the lowest class, there were more gains in programming than everyday reasoning and no significant difference in student gains on the two forms of the assessment. In the highest class, there were more gains in programming than everyday reasoning and no significant difference in student gains on the two forms of the assessment. There was also no significant difference between the gain scores of students who started with low pretest scores and students who started with high pretest scores in both the high and low classes suggesting that participation in the robotics curriculum benefited both high and low scoring students to the same degree as measured by the assessment.

It was reported that students improved the ways they formulated a solution using given syntax, in representation, and in algorithmic thinking; and they did not improve in data processing, following a given syntax, and algorithmic thinking. Based on the results of the assessment, it was discovered that there was need to strengthen the robotics curriculum to better facilitate computational thinking in the aspects of correct syntax, data processing, and algorithmic thinking (especially parallel execution) and transfer and

connection between robotics programming and everyday reasoning. There were noted limitations of the study including test/retest effect, lack of a control group, small sample size drawn from one school, linguistic/reading challenges with the assessment, test fatigue (some students took up to an hour to complete it), and influence of the rest of the fifth grade curriculum. It was not clear at the end of the study how the robotics lessons improved or did not improve computational thinking.

A series of studies investigated the development of computational thinking in middle school students through a virtual robotics programming curriculum (Witherspoon, Higashi, Schunn, Baehr, & Shoop, 2017). The online curriculum used was developed by Carnegie Mellon University and Robomatter and involved four instructional units in robotics programming (Intro to Programming, Basic Movement, Sensors, and Program Flow) using ROBOTC Graphical programming language. It was based on an existing version of ROBOTC designed for more advanced students but used a graphical interface intended to support novices by focusing on the broader logic of programming and de-emphasizing the syntactic requirements of traditional programming languages.

The first study was a pilot study conducted in a suburban school district in southwestern Pennsylvania. A pretest was given at the end of fifth grade after completing the Basic Movement unit of the curriculum. Students were enrolled in a semester-long robotics course as sixth graders where the rest of the curriculum was taught (Sensors and Program Flow) after which an analogous post-test was administered. Significant differences were found overall between pre- and post-test scores and there were no significant differences found between male and female students on the pretest or

post-test. The expanded implementation time could have increased the effect size and it was unknown if any students participated in robotics enrichment activities over the summer break between fifth and sixth grade.

The second study involved full implementation of the curriculum in four schools across four school districts taught by four teachers that agreed to participate in the study during their regularly scheduled robotics classes with sixth, seventh, and eighth grade students. Students took a pretest, completed a period of instruction (ranging from five to seven weeks) with the online curriculum, and then completed an alternate version of the post-test. There was a small but statistically significant overall mean gain of 0.57 points from pre- to post-test. An analysis of mean gains by amount of curriculum progress was conducted of students who completed material beyond the introductory lessons ($n = 315$): Basic Movement ($n = 142$), Sensors ($n = 165$), and Program Flow ($n = 8$). There was a significant difference in pretest scores by group: small differences between the Basic Movement and Sensors group and between the Basic Movement and Program Flow groups, but no significant difference between the Sensors and Program Flow groups. ANCOVA was used with post-test scores to adjust for these preexisting differences and found significant association of progress with post-test scores. Post hoc Tukey tests revealed that students who completed only Basic Movement made no gain, students who completed Sensors made modest gains, and students who completed Program Flow made sizeable gains. It was unclear whether the curriculum caused the observed gains. Only eight students completed the Program Flow unit and the students who made it to Sensors and Program Flow had higher pretest scores from the start. It could have been any

number of things that influenced the results including student characteristics; difference in teachers; and the extent of incorporating physical robots into the virtual curriculum (this was not a requirement but was a possibility for schools with access to them and teachers who wanted to use them). Many teachers claimed the use of physical robots improved students' engagement and motivation to participate in class, and they also attributed increased learning gains to the additional practice of programming skills in a similar context using physical robots. The researchers pointed out that "even in the virtual context, robotics-specific content such as sensor function and motor port location may direct students' attention away from programming skills and toward more mechanical aspects of the robot's operation, and add additional time to the curriculum without necessarily increasing their exposure to computational thinking content" (Witherspoon et al., 2017, p. 15).

Another study used a shortened version (three levels: Basic Movement, Sensors, Program Flow) of the same curriculum and attended to how structural programming features predict differences in learning and motivation (Witherspoon, Schunn, Higashi, & Shoop, 2018). The version of the curriculum used in the 2017 study (Witherspoon et al., 2017) was revised for this study so that each unit was shortened by removing sections that did not contain conceptual programming content with the intent of allowing more students to reach later units. The curriculum could be completed in the virtual environment but was designed to replicate physical robotics hardware and there was the capability to download and test programs on VEX IQ robots for teachers with access to them. This study looked at 136 sixth and eighth grade students within seven different

robotics classes taught by three teachers (each teacher taught two to three sections of robotics using the virtual curriculum) across two schools in southwestern Pennsylvania. Participants in the study completed different levels of the curriculum: 39 finished only the Basic Movement unit (M_{age} = 11.42), 40 completed Basic Movement and Sensors (M_{age} = 11.26), and 57 completed all three units (M_{age} = 13.11). A 25 item programming assessment targeting robot programming, general programming, and computational thinking was given as a pretest and post-test, as was an attitude survey containing 12 items asking about interest, competency beliefs, and identity in programming.

Results showed significant learning gains overall with increases of about eight percentage points from pre- to post- for all groups and larger gains for the Sensors and Program Flow groups. A further breakdown of scores was available. Significant gains were reported in the following areas: robot programming (Sensors group); general programming (Sensor and Program Flow groups); and computational thinking (Program Flow group).

There was a decline in motivation from pre- to post- for all three groups but to different degrees. There was a significant decline in interest from pre- to post- for students in the Basic Movement group, smaller marginally statistical declines for the Sensors group, and no significant decline for the Program Flow group. Only the Sensors group showed small but marginally significant pre-post differences in Identity. Significant declines in Competency Beliefs were found for both the Basic Movement and Program Flow groups.

When analyzing the results by gender, there was no difference in motivation between males and females, and females outscored males on every section of the programming assessment translating to girls performing about fourteen percentage points higher than males on the Robotics Programming items, about twelve percentage points higher on the General Programming items, and about nine percentage points higher on the Computational Thinking items, when controlling for pretest scores.

There was lack of random assignment in this study so it was uncertain whether differential exposure to the curriculum caused the observed changes. Other factors like instructional approach or use of physical robots may have varied. There was also an age confound with older students advancing to the Program Flow. Other factors could have also had an influence including different exposure to other curriculum (e.g. mathematics).

While the previous middle school studies used a virtual robotics curriculum with possible varied supplementation with physical robots, other studies have intentionally manipulated and examined these differences. One such study was conducted in four urban eighth grade middle school classrooms across two public schools in Chicago comparing two-week curricular units (Berland & Wilensky, 2015). The researchers designed VBOT software and provided four activities (five days) to four classes – two that programmed virtual robots (one class at each school) and two that programmed physical (LEGO Mindstorms) prebuilt robots (one class at each school). There were two different male teachers at one school and one female teacher at the other school.

Results revealed that both groups improved about the same amount but developed different perspectives on the content: The groups that worked with physical robots interpreted situations from the bottom-up "agent" based perspective (i.e. they focused on their individual robot and how it might affect the system). The group that worked with the virtual robots used the top-down "aggregate" perspective (i.e. they focused on the whole system and its impact on individual robots). The Virtual group produced more circuits (only 20%) that worked well in context because they were constantly on the computers, while the Physical group would stop to test their programs with the robot. The Physical group adapted their circuits to imprecise motor function, considered the physical mass of the robot, and used mass and speed in their circuit design which was not simulated in the virtual setting. The Physical group also created more unique, difficult, and dense circuits and more complex individual robots. They created more complex and difficult circuits and spent more time working on each circuit. The researchers suggested that time, material and environment matter "far less" than motivating students to share and tinker, and tinkering, playing and sharing allowed students to understand a complex set of content better and in a short period of time (Berland & Wilensky, 2015).

The trend of using LEGO Mindstorms robotics kits continued in other studies. A study of junior high (age 15) and high school (age 18) students in Greece used LEGO Mindstorms NXT 2.0 educational robotics kits in a series of robotics training seminars (Atmatzidou & Demetriadis, 2016). A total of eight seminars (four at the junior high and four at the high school) made up of 11 sessions (two hours each/once weekly) were led by trained postgraduate students with regular teachers in the classroom to help maintain

lesson flow. A questionnaire to assess the level of computational skills development was administered at the end of the fourth session and again at the end of the seminar. A 'think-aloud' protocol and student opinion questionnaire were also administered at the end of the seminar.

The model applied in the study focused on five dimensions of computational thinking: abstraction (the process of creating something simple from something complicated); generalization (transferring a problem-solving process to a wide variety of problems); algorithm (a practice of writing step-by-step specific and explicit instructions for carrying out a process); modularity (the development of autonomous processes that encapsulate a set of often used commands performing a specific function that might be used in the same or different problems); decomposition (the process of breaking down problems into smaller parts that may be more easily solved). Results revealed that students reached the same level of computational skills development independent of age and gender; computational thinking scores improved significantly toward the end of the activity (demonstrating the need for sufficient time to develop); gender relevant differences appeared when analyzing scores in specific dimensions of computational thinking skills; girls appeared to need more training time to reach the same skill level compared to boys; and the modality of the skill assessment instrument may have impacted performance (oral versus written; boys were more reluctant to write than girls). Students reported that working with robots not only helped them develop deeper understanding of programming, but also kept them interested and motivated to keep working on programming. Limitations of the study included lack of a control group, lack

of a pretest, and exclusion of the high school group in gender analyses due to uneven distribution in the sample (i.e. number of boys far exceeded number of girls).

One computational thinking study that used a commercially available robot other than LEGO Mindstorms studied the mBot robot that is programmed using mBlock, which is a visual programming language similar to Scratch (Fronza et al., 2017). Two teams of six to seven girls and one boy in the ninth grade in a liberal education environment participated in a course (ten hours over three days) to teach computational thinking concepts using educational robotics. The goal was to create an algorithm for the robot to solve a maze. Neither group succeeded with the goal; instead they hand coded a solution specific to the maze that did not work properly though the authors claimed that student logic was correct. The authors used code analysis to claim that students used the specific computational thinking concepts of sequences, loops, events, conditionals, operators, and data. Code analysis can be a problematic way to analyze and infer results and there was no formal assessment of learning in the study.

Another study moved away from the use of commercially available robotics and used a robot and software developed by the researchers. The robot was built by the researchers and paired with an app that they developed using Arduino on the iPad. They added a Bluetooth HM-10 BLE component to the robot so that it could communicate with the iPad (Phetsrikran, Massagram, & Harfield, 2017). They tested it at Saint Nicholas School in Phitsanulok, Thailand with 20 students aged 13-14. There were four groups of five with one observer per group. A short introduction (less than five minutes) was given of how to use the robot and send commands to it from the iPad. Then each group was

given a robot and set of puzzles (easy to difficult) and played and solved puzzles for two hours. Everyone gathered to share their feelings and experience at the end of the session.

Positive feedback from the students included an interest in the robots, enjoyment writing programs to control the robots, and satisfaction solving easy and medium puzzles. Negative feedback from students included the need to precisely position the robot at the starting point, the problem of the robot "not walking straight", the mobile app freezing or crashing, and the difficulty of the puzzles. Positive feedback from the observers included the majority of students ending the session with improved computational thinking skills and thinking logically to solve the puzzles as well as being motivated enough to keep solving puzzles up to and beyond the hard level. Negative feedback from the observers included some concepts needing more time or being more challenging for students to understand; one group had students who did not play equally; and particular students spent more time on the iPad than others.

The authors claimed that although there was no assessment, there was evidence of practical skills: persevering – evidenced by complaints of puzzle difficulty; tinkering and debugging – evidenced by repeated attempts to complete puzzles in the minimum number of steps; and collaboration – evidenced by interaction between students to solve puzzles. While this study provides some valuable information, there was no formal assessment used, just observation and self-report, and the tools used are not widely and readily available. There are also only four puzzle levels; what comes next after all of the puzzles are solved?

The researchers did observe the advantages of combining mobile technology and educational robotics including better social interaction (the focus of attention was on the puzzle and robot instead of staring at a computer), enhanced collaboration (more students can sit around the iPad and pass it around to work on puzzles), increased portability (does not require a computer lab – worked well in the regular classroom), and natural user interface (drag and drop is well suited to the touchscreen on mobile technology). These benefits could apply to any tools that combine mobile technology and educational robotics, thus finding a more widely available app and robot may make widespread implementation a more realistic possibility.

Commercial robots that pair a robot with an app on a mobile device are beginning to appear. Ozobots are small robots (about 2.5 cm in height and diameter) that use online block-based visual programming for the coding (Merino-Armero et al., 2018). Merino-Armero et al. (2018) investigated map reading skills in third grade with an interest in computational thinking and motivation. They tested computational thinking using the Computational Thinking Test, but did not report the results. The control group used paper and pencil, and the experimental group used Ozobots. The ARCS (Attention, Relevance, Confidence, and Satisfaction) Model (Keller, 1987, 2010) was used to investigate motivation in the study and was measured using the Instructional Materials Motivation Survey (IMMS) (Keller, 2010). In this model, confidence is a goal that should be present for motivation to learn, and it happens if learners believe they will be successful carrying out the task or learning (Merino-Armero, González-Calero, Cózar-Gutiérrez, & Villena-Taranilla, 2018, p. 184).

A significant main effect of group on motivation was found; students in the experimental group showed higher motivation scores than the control group. There were also significant differences in motivation depending on gender; males had higher motivation levels than females. When investigating the results for each of the four dimensions of ARCS, there was a significant main effect of the intervention group with the experimental group obtaining better results in attention, relevance, and satisfaction than the control group. While there was no significant effect of gender on attention and relevance, there was a significant main effect of gender on satisfaction with males showing higher satisfaction rates. There was a non-significant main effect of the experimental group on confidence, but the average score for the experimental group was higher than the control group. There was a significant effect of gender on confidence with boys showing higher scores than girls. The intervention length was two hours, so Merino-Armero et al. (2018) concluded that the inclusion of computational thinking with educational robotics is highly motivating in the short term and they recommended that future studies address whether improvements in motivation using educational robotics to introduce computational thinking improves learning.

The use of commercially available robots increases the likelihood that studies can be replicated in a wider variety of settings. Many of the concerns noted in Benitti's 2012 systematic review (and other reviews conducted since then) including mixed results, methodological concerns (e.g. lack of control group and formal assessment) and an overabundance of studies using LEGO robotics continue today. While the appearance of

the specific robots used in the reported studies may vary slightly, in general the robots share a machine-like appearance (e.g. blocks with wheels and sensors).

The K-12 version of the latest New Media Consortium (NMC) / Consortium for School Networking (CoSN) Horizon report (Freeman, Becker, Cummins, Davis, & Hall Giesinger, 2017) presented educational robotics as one of the most important advances in technology in the short term, due to the enormous diversity of possibilities it offers (Merino-Armero et al., 2018, p. 186). There is a lack of studies that analyze the integration of robotics in the classroom (Benitti & Barreto, 2012; Poh et al., 2016) but it does seem to be beneficial in many areas (e.g., Hong, & Chen, 2014; Karim et al., 2016; Lindh & Holgersson, 2007; Poh et al., 2016). A review of robot use in K-12 STEM education found that many studies reported the positive role that robots play in several areas including learning of educational activities; developing creative thinking; improving problem-solving skills; and increasing motivation, engagement, and attitude towards education (Karim et al., 2016, p. 6). Robots in education can be used as a teaching tool, can facilitate learning, and can support the development of 21st century learning skills (Cheng et al., 2018). Programmable robots can allow children to test the robots' actions in the environment as well as their own reasoning strategies (Caci, D'Amico, & Cardaci, 2004). Robots make it easier to find and fix errors sooner because most of the time they provide direct evidence of solution accuracy (Fronza, El Ioini, & Corral, 2017).

Variety is lacking in the type of robots that have been empirically studied. Papert anticipated that the development of new technological tools could provide new

educational opportunities. The emergence of commercially available emotional-educational robots creates a new opportunity for research to fill a gap in the literature that did not previously exist.

Technology, Emotions, and Learning

Studies of children's views of robots can provide valuable insight into how students may interact with selected robots. Several studies have been conducted in this capacity. In one such study, children ages nine to eleven were given five robot images (a total of 40 images were viewed in eight groups) to rate (Woods, Dautenhahn, & Schulz, 2004). The robot images generally fell into three categories: human-looking robots, machine-looking robots, and animal-looking robots. Children rated human-looking robots as having feelings and being able to understand them. They also rated animal-looking robots as being able to understand them. Human-like robots were rated with the highest negative behavior scores (meaning that the children perceived them as the most aggressive or bossy). A combination of animal-like or machine-like robots was perceived as the friendliest and shyest, and children rated pure animal-like robots as being happier than pure machine-like robots.

Robots with the following characteristics were rated as having negative behaviors: legs or wheels, rectangular body, machine-like appearance, male gender, realistic appearance, less exaggerated facial features or partly camouflaged face, and dull colors. Positive behaviors were associated with robots having the following characteristics: facial features, gender (male or female), cartoon appearance, exaggerated facial features

(especially eyes), using legs or wheels to move, bright colors. Children aged nine to eleven assigned genders to the robots, especially male gender, and robots that were assigned female gender were associated with positive traits like happiness and friendliness. Generally, the children judged the human-like robots as aggressive, but human-machine robots as friendly which the authors suggested provided support for the Uncanny Valley. The idea of Uncanny Valley was "described by Mashiro Mori who contended that if robots become too close to realism (appearing very human), but are not perfect (indistinguishable from humans), then the imperfections can be viewed extremely negatively" (Woods et al., 2004, p. 47).

It was noted that some of the children who participated had seen the images before the study, and this may have influenced their perceptions of the robots. The authors also noted the need for behavioral data (children interacting with physical robots) to confirm the findings because relying on photographs makes it hard to set context and relate appearance to actual behavior and interaction. While this study used only photographs, another study did investigate the interactions of children with different kinds of technological agents.

Twenty-six three to ten year old children interacted with Amazon Alexa, Google Home, Cozmo, and Julie Chatbot; and then an interactive questionnaire was given to gather their perceptions (Druga et al., 2017). Most of the children agreed that the agents were friendly and trustworthy, and they used gender interchangeably when talking about them. Some of the children were not sure whether Cozmo was a boy or a girl and one boy concluded that Cozmo was a bobcat with eyes. The three and four year olds very

much enjoyed playing with Cozmo. When investigating interactive engagement, the researchers found that although the children were attracted to the voice and expressions of the agents at first, they lost interest when the agent could not understand their questions. Two of the children, Gary and Larry, liked interacting with Cozmo the most because Cozmo can move and has feelings. "Cozmo was able to effectively communicate emotion, and so the children believed that Cozmo had feelings and intelligence" (Druga et al., 2017, p. 599).

Affective computing studies (Picard, 2003; Tao & Tan, 2005) are helping designers develop computers that can sense, recognize, and express emotions (Triberti et al., 2017). Designers can intervene on the aesthetic appeal of interfaces to promote positive feelings in users (Triberti et al., 2017). Studies in multimedia learning showed that embedding emotional stimuli into interfaces could elicit positive emotions in learners and improve learning outcomes (Plass, Heidig, Hayward, Homer, & Um, 2013; Um, Plass, Hayward, & Homer, 2012). It has been suggested that even though there has been increased interest in studying emotions in education, affective computing and learner emotions have not yet become a serious research interest among educational technologists (Akbiyik, 2009; Schutz & DeCuir, 2002).

There is a research area called affective robotics that deals with robots simulating emotions and other human expressions and body language. The goal in this field is to design robots that interact with their users in natural ways (Demir, 2017; Sullins, 2008). Giving robots the ability of emotion expression is a major research topic in robotics (Huang, Yun, Yujin, Changlin, & Lianqing, 2018). Huang et al. (2018) explained that

using a robot as the learning media conforms to cognition theory as well as the interactive cognitivism philosophy which emphasizes that cognition comes from interaction.

Rapid progress in robotics makes naturalistic interaction between humans and machines an everyday reality. Advances in sensors, computer vision, audio processing, and machine learning, are enabling robots to understand their environment and their users better, and, in turn, to become more adaptive and flexible, responding better and engaging a broader variety of tasks such as household chores, tourists guiding, education, autism research, healthcare, etc. (Pantic, Evers, Deisenroth, Merino, & Schuller, 2016, p. 1477).

It has been well established that emotions affect learning (Akbiyik, 2009; Scherer, 2005) and that emotion can be a powerful force in enhancing or inhibiting learning (Picard, Vyzas, & Healey, 2001). Ormrod (2012) explained that learning is both an affective and cognitive enterprise, and that teachers should make sure students' learning experiences are positive and nonthreatening. Emotionally sound instruction when using educational technology includes the use of instructional strategies to increase positive emotions and decrease negative emotions (Astleitner, 2000). There is evidence showing that emotions play a fundamental role in perception, learning, attention, memory, and other abilities (D'Amico & Guastella, 2018).

Emotional robots are finding their way into the consumer market. Papert (1980) advocated for investigating the tools that a culture provides and their potential impact on

student learning. A new question emerges as a result of the development of new

technologies: How might an emotional-educational robot impact student learning?

Cozmo

There is call for studies of emotional-educational robotics (Kwak, Kim, Kim,

Shin, & Cho, 2013). The challenge comes in finding an emotional-educational robot that

is appropriate for use with children. Cozmo is an artificially intelligent robot with

computer vision, animation, and emotions that is marketed as a toy for ages eight and up.

Cozmo has been described as "lifelike enough to evoke sympathy, but still enough of a

toy not to teeter too close to the Uncanny Valley" (Statt, 2016). A former Pixar animator

co-designed Cozmo with inspiration coming from lovable robots seen in animated films

like "Wall-E" and influences including Wall-E, The Iron Giant, and Astro Boy (Statt,

2016). Hanns Tappeiner, Anki's co-founder and president, explained that "Cozmo

doesn't just move through his world – he can manipulate it. People perceive

manipulation as intelligence" (Statt, 2016).

> While we can easily argue that this robot toy can only mimic emotional behavior
>
> determined by a computer program, the imitation is quite fascinating. For
>
> example, Cozmo presents behavior imitating happiness when it wins a game.
>
> Cozmo seems to get scared when it is nearly falling from an edge of a table.
>
> According to Cozmo designers, the robot is able to present behavior related to
>
> tens of emotions. (Demir, 2017, p. 161)

In a review of Cozmo, Statt (2016) highlighted many of its features. Cozmo uses facial recognition technology (powered by a camera in the mouth location on its face) to remember different people. Artificial intelligence allows Cozmo to learn and adapt to the user over time. Cozmo is small, lightweight (just 150 grams) and very portable thanks to a mobile app that handles the heavier processing tasks and is paired with the robot over Wi-Fi. Cozmo comes with three sensor-embedded blocks that can be used to play games with the robot and to help it understand its position in the environment. The built-in accelerometers in the blocks can sense taps or motion, and control color LEDs (Touretzky, 2017). The cubes communicate with the robot using a proprietary radio protocol and they include special markers that allow Cozmo to visually detect them (Touretzky, 2017). Cozmo can lift, carry, roll, and stack the cubes.

One of the standout features of Cozmo is the inclusion of what Anki calls an *emotion engine* that powers a wide range of different states the robot is capable of emulating. Drawing from academic psychology, those different states – happy, calm, brave, confident, and excited, to name a few – are derived from combinations of the big five personality traits used to describe the human psyche. By mixing and mashing these traits as if they were colors, Cozmo can replicate a complex range of human-like emotions (Statt, 2016).

Cozmo's animators designated ranges for qualities like how fast and high Cozmo raises its lift, moves its head or eyes, or expresses something using sound. According to Tappeiner, when you play animations multiple times, they feel canned, but the ranges

allow Cozmo's artificial intelligence to make those decisions on its own so there is no sure way to determine how Cozmo will react to any given situation. (Statt, 2016).

Cozmo's software covers navigation, object recognition, manipulation, and complicated programming and gives users direct access to its capabilities (Gorman & Ackerman, 2017). Cozmo was found to be accessible, usable, and flexible even for novice programmers (Gorman & Ackerman, 2017; Kennington & Plane, 2017). Cozmo has arms that can lift or push small objects, track wheels for movement, a text-to-speech synthesizer, and a camera embedded in a small movable head that has animated eyes (Kennington & Plane, 2017). The size and affordances of the robot make it manageable for researchers who are not roboticists, and Cozmo is affordable (under $180) and very portable (Kennington & Plane, 2017). Cozmo is controlled by more than 1.6 million lines of code, but when combined with Scratch Blocks, programming Cozmo becomes accessible and more like playing a game; in fact, in app stores, Cozmo is not listed under programming, it is listed under games (Diwanji, 2017). Cozmo is marketed as a toy for children to learn procedural 'coding' and in the observations of Kennington and Plane (2017), children enjoyed interacting with Cozmo and found Cozmo aesthetically pleasing. Touretzky (2017) echoed this observation: "Cozmo is a complex device with interesting behavior that children actually care about" (p. 78). "It may seem like a small children's toy, but in fact, it has one of the most sophisticated AI back-ends that has ever made its way to the consumer's domain" (Akimana, Bonnaerens, Wilder, & Vuylsteker, 2017, p. 8).

Cozmo's Software Development Kit (SDK) even provides a resource for people to tap into robotics and AI if they are interested. Thanks to a partnership between Carnegie Mellon University and the Montour School District, students at the district's middle school used Cozmo in the first K-12 artificial intelligence curriculum in the United States in the fall of 2018. Dave Touretzky, a professor in the computer science department at Carnegie Mellon specializing in cognitive robotics and computer science education, helped train teachers for the project. According to Touretzky:

> Unlike many other robots used to teach coding in classrooms across the county, Cozmo can see. Even before students start experimenting with writing their own code, the robot is capable of interacting with the user and its surroundings. It's a toy powered by artificial intelligence. (Martines, 2018, p. 4)

Kinvert is a STEM-focused educational company that uses Cozmo to help people learn coding and robotics (Kaiser, 2018). According to Keith Young, the owner of the company (who holds degrees in both mechanical and aerospace engineering):

> Most run-of-the-mill STEM robots offer nearly zero technical depth. We don't teach with some of the most popular STEM robots for this reason. In my opinion, educators need access to sensors which can drive meaningful decisions in code. Unfortunately, with few exceptions, the most meaningful programming available among other robots is line following and turning when sensing a color. Comzo is in a different league in many ways, from personality to student engagement to software potential. (Kaiser, 2018, paragraph 13)

Young reports that Cozmo is great for STEM events because of how engaging it is, and they have actually noticed an increase in female sign-ups for their robotics course overall in number and percentage of the full roster since they began using Cozmo (Kaiser, 2018).

Georgia Tech students enrolled in Introduction to Perception and Robotics are using Cozmo by learning to apply fundamental programming algorithms to an actual robot (Giles, 2017). The instructor of the course reported that the previous platform was not as exciting and interactive as Cozmo, so the human-robot interaction was typically limited to only laboratory assignments, but students report playing with Cozmo even when they are not in class and Cozmos can be spotted all over campus (Giles, 2017). One student reported:

> I knew nothing about applying code to an actual robot. I love seeing the robot move and knowing I completed the programming that makes it navigate. This is the first time I've been able to apply skills I've learned in other classes and see the results in action. (Giles, 2017, paragraph 7)

The instructor reported that "for many computing majors, robotics provides a way to see code transferred into something tangible that has an effect on the physical world – even if the tangible object is only a small robot" (Giles, 2017, paragraph 11).

Anki released the Python Software Development Kit (SDK) to allow programmers to control the robot, and they opened that capability to children and non-

programmers in 2017 with the release of Code Lab (Diwanji, 2017). Code Lab is based

on Scratch Blocks which makes it the first toy built for children using the program.

Code Lab offers two different modes of interaction. Sandbox Mode features

horizontal grammar (see Figure 9). Horizontal grammar uses icons and the blocks snap

together horizontally, the way a child learns to read, making it similar to ScratchJr which

was developed for children ages five through seven. Constructor Mode adds a layer of

vertical grammar (see Figure 10). Vertical grammar is text-based and the blocks snap

together vertically, making it similar to Scratch which was developed for eight through

sixteen year olds but can be used by people of all ages.
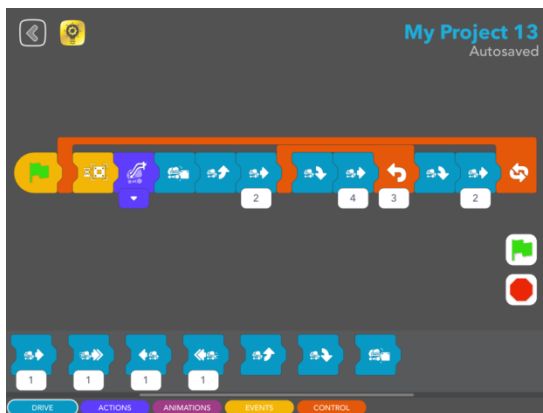
Figure 9. Cozmo Code Lab Sandbox Mode. Retrieved from

https://developer.anki.com/blog/news/cozmo-code-lab/.
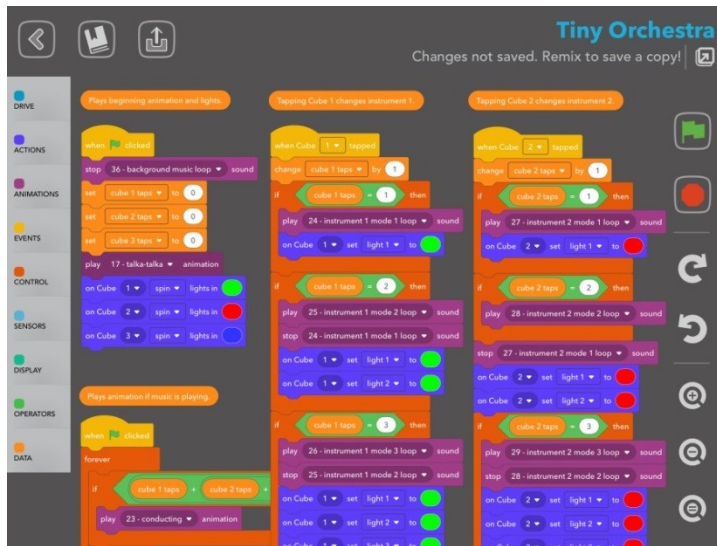
Figure 10. Cozmo Code Lab Constructor Mode. Retrieved from

https://developer.anki.com/blog/news/cozmo-code-lab/.

Cozmo's Code Lab offers novices the opportunity to learn coding and

programming concepts in a meaningful way with a robot that offers artificial and

emotional intelligence that is unparalleled in the current world of educational robotics.

The current study will investigate both cognitive (computational thinking and spatial

skills) and affective (competency beliefs and engagement) learning outcomes with the

educational emotional robot Cozmo.

Computational Thinking

Researchers advocate that computational thinking is at the core of all STEM

disciplines and intrinsic to every other discipline (Henderson et al., 2007; Wing, 2006).

For instance, Swaid (2015) showed a direct relationship between STEM and

computational thinking, and programming and coding are understood to be an important

part of STEM (Weintrop et al., 2016). While computational thinking is not programming, programming is a part of computational thinking (Resnick et al., 2009).

There have been several proposals for how to integrate programming and computational thinking (CT) in the K-12 curriculum (Angeli et al., 2016; Barr & Stephenson, 2011; Brennan & Resnick, 2012; Dettori et al., 2016). Some suggest an interdisciplinary approach (Angeli et al., 2016) while others recommend a specific subject area (Brennan & Resnick, 2012). Educational policies have been aimed at introducing computational thinking into the curricula (Manches & Plowman, 2017; Shute et al., 2017; Swaid, 2015) and yet there remains a lack of consensus on a formal definition of computational thinking (Román-González et al., 2017), how to measure it (S. Grover & Pea, 2013), and disagreement over how to integrate it into the curricula (Lye & Koh, 2014).

Papert may have been the first to use the term computational thinking in 1980 when describing a mental skill children develop from practicing programming (Denning, 2017), but its meaning has developed over time:

The meaning of computational thinking, evolved since the 1950s, is clear and supports measurement of student progress…An algorithm is not any sequence of steps but a series of steps that control some abstract machine or computational model without requiring human judgment. Computational thinking includes designing the model, not just the steps to control it. Computational thinking is loosely defined as the habits of mind developed from designing programs,

software packages, and computations performed by machines. (Denning, 2017, p. 33)

Aho (2011) proposed that abstractions called computational models are at the heart of computational thinking and that the process of computational thinking is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. Denning (2017) suggested that the computational models described by Aho are not limited to computer science and can refer to any mode in any field that represents or simulates computation.

Denning (2017) suggested that the current surge of interest in computational thinking began in 2006 when Jeannette Wing, then a National Science Foundation assistant director for Computer & Information Science & Engineering (CISE), called for a mobilization of resources to bring it into K-12 schools. "The result was a vague definition that targeted not only designers but all users of computational tools, anyone engaging in step-by-step procedures, and anyone engaging in a practice that could potentially be automated" (Denning, 2017, p. 38).

Wing's (2006) definition of computational thinking (CT "involves solving problems, designing systems, and understanding human behavior") drew criticism from some researchers including Glass (2006) who posited that the description was so close to the meaning of problem solving that computational thinking did not exist. Wing updated the definition in 2010 in collaboration with other researchers to include "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" (Cuny, Snyder, & Wing, cited in Wing, 2010, p. 1).

Denning (2017) noted several criticisms of Wing's newer definition of computational thinking including: the absence of any mention of computational models; the suggestion that any sequence of steps constitutes an algorithm (the steps are not arbitrary and must control some computational model); and unsubstantiated claims (e.g., everyone can benefit from learning CT, users of computational tools will develop CT, CT will help people perform everyday procedural tasks better, CT enhances general cognitive skills that will transfer to other domains where they will manifest as superior problem-solving skills) that "undermine the effort by overselling computer science, raising expectations that cannot be met, and leaving teachers in the awkward position of not knowing exactly what they are supposed to teach or how to assess whether they are successful" (p. 34). Tedre and Denning (2016) asserted that the claim of automatic skill transfer from computational thinking to different knowledge domains was debunked in the 1980s (R. D. Pea & Kurland, 1984) and more recently in a book that reaffirmed the lack of evidence to support the claim (Guzdial, 2015, pp. 27-29, 39-40) (p. 121). Denning (2017) concluded that computational thinking primarily benefits people who design computations and claims of benefit to non-designers are not substantiated.

The essence of computational thinking involves breaking down complex problems into more familiar/manageable sub-problems (problem decomposition); using a sequence of steps (algorithms) to solve problems; reviewing how the solution transfers to similar problems (abstraction); and finally determining if a

computer can help us more efficiently solve those problems (automation). (Yadav, Hong, & Stephenson, 2016, p. 565)

Shute, Sun, and Asbell-Clarke (2017) defined computational thinking as the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without computers) with solutions that are reusable in different contexts. Shute et al. (2017) suggested that computational thinking "is an umbrella term containing design thinking and engineering (i.e., efficient solution design), systems thinking (i.e., system understanding and modeling), and mathematical thinking as applied to solving various problems" (p. 5). They proposed that using computational tools appears to increase computational skills because of their close relationship with computing and programming, and that robotics has been fruitful for developing computational skills because of its close relationship to programming. Both programming and robotics emphasize various components of computational thinking. In programming the learner analyzes the problem, breaks it into constituent processes, writes programs that require abstraction and generalization, and tests program correctness and efficiency when debugging (Shute, Sun, & Asbell-Clarke, 2017). Robotics provides learners with tactile experiences to solve problems using computational thinking skills including identifying the problem/goal for the robot, decomposing the problem, developing an algorithm for the robot to follow instructions and act accordingly, and the iterative processes of systematic testing and modifying required by debugging when it doesn't work properly (Shute et al., 2017).

Barr and Stephenson (2011) proposed a definition for computational thinking across all K-12 education. They mapped a variety of conventional school subjects onto a list of computational thinking concepts. Weintrop et al. (2016) proposed a definition of computational thinking for math and science using a taxonomy with four main categories: data practices (collecting, creating, manipulating, analyzing, and visualizing data); modeling and simulation practices (using computational models to understand a concept and find and test solutions, and assessing, designing, and constructing computational models); computational problem solving practices (preparing problems for computational solutions, programming, choosing effective computational tools, assessing different approaches/solutions to a problem, developing modular computational solutions, creating computational abstractions, troubleshooting and debugging); and systems thinking practices (investigating a complex system as a whole, understanding the relationships within a system, thinking in levels, communicating information about a system, defining systems, and managing complexity). They suggested that computational thinking can deepen learning of math and science and that math and science provide a meaningful context to apply computational thinking which adds to its authenticity because students can get a sense of real-world applicability.

> For the purpose of conceptualizing CT and integrating it in education, we should not try to give an ultimate definition of CT, but rather try to find similarities and relationships in the discussions about CT. Finding these similarities and relationships will lead to a more concise description of "what matters" in CT and how to integrate it within K-12. (Voogt et al., 2015, p. 726)

There have been many frameworks suggested for computational thinking. For example, the Computer Science Teachers Association issued an operational definition in 2011, the Computing at School subdivision of the British Computer Society followed in 2015 with a more detailed definition, and the International Society of Technology in Education followed in 2016 with a generalized technology definition (Denning, 2017).

Brennan and Resnick (2012) developed a framework for computational thinking with three dimensions. The first dimension is computational concepts which include:

- sequences – specifies the behavior or action that should be produced

- loops – a mechanism for running the same sequence multiple times

- events – one thing causing another to happen (produces an action)

- parallelism – sequences of instructions happening at the same time

- conditional – the ability to make decisions based on certain conditions (supports the expression of multiple outcomes)

- operators – provide support for mathematical, logical and string expressions (enables the programmer to perform numeric and string manipulations)

- data – involves storing, retrieving, and updating values

The second dimension is computational practices that focus on the process of thinking and learning and move beyond *what* is learned to *how* it is being learned. This dimension includes:

- experimenting and iterating – an adaptive process where the plan might change in small steps in response to approaching a solution

- testing and debugging – developed through trial and error, transfer from other activities, and support from knowledgeable others

- abstracting and modularizing – building something by putting together collections of smaller parts

The third dimension is computational perspectives and includes:

- expressing – design and self-expression

- connecting – creating with and for others

- questioning – interrogating the taken-for-granted and in some cases responding to that interrogation through design

Seiter and Foreman (2013) proposed the Progression of Early Computational Thinking (PECT) model which assumes that every student has a latent (not directly observable) proficiency in computational thinking that manifests itself in the student's ability to design and implement programs to complete specific tasks. The model is made up of three fundamental components of decreasing abstraction: Computational Concepts (procedures and algorithms, problem decomposition, parallelization and synchronization, abstraction, data representation), Design Pattern Variables (these are based on common coding patterns in Scratch and include animate looks, animate motion, conversate, collide, maintain score, user interaction), and Evidence Variables (ranked characteristics of Scratch code including looks, sound, motion, variables, sequence & looping, Boolean expressions, operators, conditionals, coordination, user interface event, parallelization, initialize location, initialize looks). The researchers explained that Evidence Variables and Design Pattern Variables are assessed using a rubric (1=Basic, 2=Developing,

3=Proficient) based on their experience teaching Scratch programming to K-12 students, and that measuring Computational Concepts is outside the scope of their research (Seiter & Foreman, 2013).

García-Peñalvo and Mendes (2018) discussed several additional computational frameworks and concluded that computational thinking is not coding, but computational thinking may be the outcome of good planned programming practice. "Most teachers and education researchers have the intuition that computational thinking is a skill rather than a particular set of applicable knowledge" (Denning, 2017, p. 36). However, there is no consensus on what constitutes the skill and current assessment methods are unreliable indicators (Denning, 2017). A skill is an ability acquired over time with practice – not knowledge of facts or information; thus students have been tested on their knowledge but not their competence (Denning, 2017). The idea of assessing skill by performance (e.g., code-a-thons and projects that assess skill by performance) is becoming more prevalent, and Denning (2017) suggested that students would benefit if computational thinking is approached and assessed as a skill.

Brennan and Resnick (2012) described different approaches for assessment including project portfolio analysis (e.g., using the tool Scrape that analyzes the programming blocks in Scratch projects); artifact-based interviews; and design scenarios. They acknowledged that no single approach is sufficient and that a combination of approaches might be appropriate. There is a lack of tests relating to computational thinking that have undergone a comprehensive psychometric validation process (Román-González et al., 2017), but there are a few that have been reported in the literature.

Dr. Scratch is a free, open source web application designed to automatically analyze Scratch projects and provide feedback (Moreno-León, Robles, & Román-González, 2015). Its convergent validity with respect to other traditional metrics of software quality and complexity was reported by Moreno-León, Robles, and Román-González (2016). One test aimed at middle school students that has been subjected to psychometric requirements and undergone a validation process is the Test for Measuring Basic Programming Abilities (Mühling, Ruf, & Hubwieser, 2015). The Computational Thinking Test is another psychometrically validated instrument designed for students in grades five through ten that can be used for collecting quantitative data in pre-post evaluations to determine the efficacy of programs or curricula aimed at fostering computational thinking (Román-González et al., 2017). Román-González et al. (2017) pointed out that the practice of using the assessment to collect quantitative data is desirable since the qualitative approach has been used the most throughout the literature. This instrument will be used in the current research study to measure computational thinking skills.

Spatial Skills

The definition of spatial ability is a matter of debate, and a comprehensive account of underlying processes is not available (Hegarty & Waller, 2005). There is less consistent support for factors such as *spatial orientation* which involves the ability to imagine oneself or a configuration from different perspectives (Hegarty & Waller, 2005). Sutton and Williams (2007) suggested that spatial ability is defined as the performance on tasks that require mental rotation of objects; the ability to describe and understand

how objects appear at different angles; and an understanding of how objects relate to each other in space. In a meta-analysis conducted by Uttal et al. (2013), agreement was strongest for the existence of a skill called *spatial visualization* which involves the ability to imagine and mentally transform spatial information.

Analyses have reported that spatial abilities can predict STEM attainment and achievement (Wai, Lubinski, & Benbow, 2009). However, computer programming is a STEM area that is considered difficult by students and teachers alike (Armoni, 2011; Caspersen & Kölling, 2009; Gökçearslan & Alper, 2015; Nilsen & Larsen, 2011; Shadiev et al., 2014) and Askar and Davenport (2009) observed low performance in programming courses. This can lead to what Uttal and Cohen refer to as a Catch-22 because early on some students "do not yet have the knowledge that would allow them to succeed despite relatively low spatial skills, and they can't get that knowledge without getting through the early classes where students must rely on their spatial abilities" (Uttal & Cohen, 2012, p. 168).

The idea that spatial abilities can predict STEM achievement assumes that spatial skills are malleable to training (Uttal et al., 2013). Many have questioned the transferability of spatial skills and argue that training is limited to cases where the trained task and outcome measure are similar (NRC, 2006; Sims & Mayer, 2002). Uttal et al. (2013) conducted a meta-analysis of three types of spatial skills training studies (video games, course, and spatial training task). They found that spatial skills are generally moderately malleable, and that overall, each program produced significant positive improvement in spatial skills. An effect size of 0.48 indicated an improvement of almost

one half a standard deviation on transfer tasks. Participants with lower levels of performance in the beginning improved more in response to training than those with higher levels initially (Uttal et al., 2013).

Students can face many spatial challenges when attempting to learn computer programming. When programming with objects, giving directional commands can be a problem for students. For example, students failed to give appropriate directional commands to characters in the code.org website (Kalelioglu, 2015). Similarly, students had difficulty determining which command could move the ladybug in the appropriate direction using Java applets from the National Library of Virtual Manipulatives; they were searching for a button with an arrow pointing 'down' (Fesakis et al., 2013). Kalelioglu (2015) explained that the visual-spatial abilities of students can be enhanced with various activities and suggested that teachers could bring a robot to class so that students could give it commands and observe the results.

Robotics has been used in several studies in order to observe the effects on spatial skills with mixed results. Verner (2004) used RoboCell programming and a robotic arm that can be manipulated through five joints and found large gains in measures of spatial ability for both middle and high school students after practicing robotics tasks with the system. Julià and Antolí (2016) found that sixth grade students who participated in a robotics course showed a statistically significant greater increase in their spatial ability post-test mean scores compared to the increase shown by students who did not join the course. However, the difference between pre- and post-test means was not statistically significant for either group. The authors noted that sample size was very small ($n = 9$ in

the experimental group and $n = 12$ in the control group) and that the calculated p-value would be smaller with a larger sample size and thus the mean difference could be more statistically significant (Julià & Antolí, 2016). Coxon (2012) gave a stratified random sample of volunteer participants ($n = 75$) ages nine through fourteen from sixteen public schools districts' gifted programs the *Cognitive Abilities Test (Form 6) Verbal Battery* and the *Project TALENT Spatial Ability Assessments*. The experimental group ($n = 38$) then participated in a simulation of the FIRST LEGO League (FLL) Competition for 20 hours total over five consecutive days while the control group ($n = 37$) did nothing comparable. Afterward, all participants took the spatial measures again, and it was found that experimental group males evidenced significant and meaningful gains in measured spatial ability (Cohen's $d = .87$), but females did not evidence significant gains.

Spatial visualization, which involves the ability to imagine and mentally transform spatial information, is the skill most widely agreed upon as being an underlying process of spatial ability (Uttal et al., 2013). Thus, many studies use mental rotation tests as a way of measuring spatial ability. One example of a mental rotation test is the Mental Rotations Test (Peters et al., 1995) that will be used in the current study.

Competency Beliefs

There exists some debate over whether cognitive or affective influences play a more significant role in learning. For example, Lawson, Banks, and Logvin (2007) believe that reasoning ability plays a more significant role than self-efficacy in predicting STEM achievement, while Linnenbrink and Pintrich (2003) explained that learners with

high self-efficacy beliefs are more likely to be behaviorally and cognitively engaged in learning activities. It has also been reported that a person with high self-efficacy and positive outcome expectations in a specific area is more likely to dedicate time, persevere, and possibly even pursue a career in that area (Bandura, 1977; Brown & Lent, 2013). A general point of agreement for many researchers is that a person's sense of confidence – both for accomplishing specific tasks and for dealing with life in general – is an important variable influencing motivation (Ormrod, 2012).

Developing positive beliefs about programming abilities is an important motivational factor for continued participation in computer science related careers (Witherspoon et al., 2018). Robotics interventions have been widely studied in extracurricular contexts with self-selected populations. In-school robotics interventions are more likely to include students with a wide variety of interests which can make it more challenging to keep all students equally engaged in programming. Beliefs about ability in STEM disciplines can be more predictive of performance than prior experience and outcome expectations, and gifted women may be more prone to underconfidence in the traditionally male-dominated STEM fields (Pajares, 1996; Zeldin & Pajares, 2000). These findings highlight the importance of investigating student competency beliefs in STEM on a broader scale and programming abilities in computer science more specifically.

The researchers who developed the STEM Competency Beliefs survey that has been adapted for the current study assess student competency to perform in diverse

situations as well as in particular skills (Y.-F. Chen, Cannady, Schunn, & Dorph, 2017). Their conceptualization of competency beliefs can be summarized as follows:

> Competency beliefs are a core construct in social cognitive theory, defined as "people's judgments of their capabilities to organize and execute courses of action required to attain designated types of performances" (Bandura, 1986). In general, educational psychological research has revealed that competency beliefs (or self-efficacy beliefs) are an important predictor of many types of achievement behavior (i.e., choice of task, engagement, effort, and persistence) (Schunk, Pintrich, & Meece, 2008). Educational and psychological research makes a clear distinction between people's actual competence and knowledge, and their subjective judgment and perceptions of them. (Y.-F. Chen, Cannady, Schunn, & Dorph, 2017, p.1)

As increasing student engagement with computer science and programming is a goal of the current study, the motivational factors of student competency beliefs and engagement will be investigated to determine the influence that the instructional interventions may have on both.

Engagement

Fredricks, Blumenfield, and Paris (2004) define engagement as behavioral, cognitive, and emotional participation. Engagement can be conceptualized as a psychological process: "the attention, interest, investment, and effort students expend in the work of learning" (Marks, 2013, p. 154). Pugnali and Sullivan (2017) explained that

research about student engagement describes both psychological and behavioral characteristics of what it means to be engaged. Psychologically speaking, engaged students are intrinsically motivated by interest, curiosity, and enjoyment and likely want to achieve their own goals; while behaviorally speaking, engaged students demonstrate positive behaviors like concentration, effort, and enthusiasm (Brewster & Fager, 2000; Jablon & Wilkinson, 2006; Marks, 2000; Pugnali & Sullivan, 2017). The engagement scale (Chung, Cannady, Schunn, Dorph, & Bathgate, 2016) that will be used in the current study conceptualizes engagement as a person's focus, participation, and persistence on a task (Carini, Kuh, & Klein, 2006; Finn, Pannozzo, & Voelkl, 1995; Fredricks, Blumenfeld, & Paris, 2004; Fredricks et al., 2011).

Engagement highly influences both learning and motivation, and the literature is replete with examples that spotlight the importance of investigating student engagement in STEM. Levels of engagement and persistence in Computer Science may be closely associated with student perceptions of the difficulty of programming and their own abilities (Witherspoon et al., 2018). Engaging students in computer science early on gives them opportunities to apply it to other subjects and interests as they go through school (S. Grover, 2014). Middle school experiences are formative for cognitive and social development in K-12; especially for future engagement with STEM fields (Tai, Liu, Maltese, & Fan, 2006). Although student engagement in school is important for achievement and social and cognitive development, studies spanning two decades documented low levels of engagement, particularly in the classroom (Marks, 2000). Marks (2000) found that engagement in academic work, unadjusted for any other

influences, decreases as grade level increases, and teachers report that student lack of engagement is a problem (Sentance & Csizmadia, 2017). Research has shown that student self-efficacy, goals for learning, and the value they assign to STEM tasks and activities are likely to influence their level of engagement (Nugent et al., 2015).

A comparison of STEM in different countries revealed that all strong STEM comparator countries worked to broaden STEM engagement and achievement by improving participation in STEM through policies that addressed both breadth and depth of learning, and covered the full spectrum of prior student achievement levels (Marginson, Tytler, Freeman, & Roberts, 2013). These policies included:

> Provision of at least some discipline-based STEM learning for all school students, up to and including students in senior secondary education; improving the engagement and performance of students from groups currently underrepresented in STEM, that on average perform relatively poorly in mathematics and science; lifting the size and average achievement of the group of students engaged in intensive STEM learning in depth, in both schooling and higher education. (Marginson et al., 2013, p. 20)

Government reviews have recognized the issues facing declining student engagement with, and participation in STEM and the objectives of national STEM legislation or policy typically include supporting increased student engagement (Marginson et al., 2013). The report recapitulated that broadening and deepening STEM

engagement is beneficial and that it is desirable to persuade more students to aspire to STEM learning and STEM-based careers (Marginson et al., 2013).

Several options are available in the search for tools to engage students in STEM and programming. Broadbent (2017) posited that people feel closer to real robots than virtual robots or computers, thus embodiment is important in human relationships with artificial technologies. The use of robots in education encourages interactive learning and improves student engagement (Highfield, 2010; Poh et al., 2016; Wei, Hung, Lee, & Chen, 2011) and the results of several studies support the claim that robots can create an interactive and engaging learning experience (e.g., Chang et al., 2010). Robotics can promote increased motivation (Chin et al., 2014; Karim et al., 2016) and Merino-Armero et al. (2018) found that even starting from an unequal initial level of motivation, there was significant improvement for both males and females who worked with educational robotics.

Robotics has been described as highly engaging (Dierking, Falk, Rennie, Anderson, & Ellenbogen, 2003; Grubbs, 2013). Robotics can be used as a tool to engage students in learning (Kim et al., 2015; Nugent et al., 2010), and empirical findings show that engagement with robotics can create a high degree of interest and engagement in STEM careers (Nourbakhsh, Hamner, Crowley, & Wilkinson, 2004; Nugent et al., 2010; Robinson, 2005; Rogers & Portsmore, 2004).

Authentic academic work involves students intellectually in a process of disciplined inquiry to solve meaningful problems of interest to them and with relevance

beyond the classroom and was found to have an influence on student engagement in a study conducted by Marks (2000). Grover and Pea (2013) suggested that curricular activities like robotics have served as means for iterative exploration of computational thinking making them ideal for motivating and engaging children.

Research question

Is there a difference in computational thinking performance, spatial abilities, competency beliefs and engagement among middle school students who are taught coding using a traditional Scratch approach versus students who are taught using the same coding activities with Cozmo, an emotional-educational robot?

Hypothesis 1

In this study, it is expected that the gains in computational thinking will be higher in the Cozmo training group compared to the Scratch training group. This hypothesis is based on empirical studies that found improved computational thinking in groups using robots compared to groups using virtual environments (Özüorçun & Bicen, 2017; Pugnali & Sullivan, 2017). Özüorçun and Bicen (2017) proposed the use of robots to facilitate algorithm logic and enhance student interest based on a learning-by-doing educational model because the robot shows students the results of writing code in the real world (Kaloti-Hallak & Armoni, 2015) and helps apply hands-on activities in the classroom. Özüorçun and Bicen (2017) found that the students placed in an experimental group that used robots for six weeks showed significant post-test changes reflecting an improvement in understanding while students in the control group (using the computer simulated

environment) showed no improvement. They explained that a robot allows students to understand how the programs work; nurtures ideas about programming robots and gives an understanding of how things work and affect their real life (Özüorçun & Bicen, 2017). The robot also encourages creativity, and helps students analyze situations and use critical thinking skills to solve real-world problems in collaboration with classmates (Özüorçun & Bicen, 2017).

Pugnali and Sullivan (2017) investigated the impact of user interface on students' computational thinking skills by directly comparing ScratchJr to a robotics platform (KIBO). The robotics group performed better on average on every Solve-It task, and also mastered sequencing and debugging significantly better than the Scratch group. The authors explained that the robots provided feedback by indicating when something went wrong, while in ScratchJr the characters will execute the program with any given programming blocks and it is up to the child to realize that what they wanted to happen in the program does not match the actions of the characters on the screen (Pugnali & Sullivan, 2017). The current study uses a parallel situation of comparing an online version of Scratch to a robotics platform. Scratch and Cozmo both use Scratch blocks as the programming platform, thus it is reasoned that robotics activities will make the results of programming more concrete and improve computational thinking compared to the computer-based simulation on the screen in Scratch.

Hypothesis 2

It is expected that any gains in spatial skills will be higher for the Cozmo group compared to the Scratch group. Sutton, Heathcote, and Bore (2005) elucidated that a substantial part of spatial ability is three-dimensional understanding (i.e., the ability to extract information about three-dimensional properties from two-dimensional representations). This requires visual and perceptual abilities to interpret what is seen, and spatial abilities to mentally manipulate visual representations (Sutton & Williams, 2007a). L. L. Liu, Uttal, Marulis, and Newcombe (2008) concluded their meta-analysis with the overall finding that although treatments may reduce the gap between male and female performance on measures of spatial ability, a gender gap persisted in most studies. Merino-Armero et al.(2018) implemented a map-reading activity to develop mental rotation skills in students using robots. The robots were used so that students could program and check their mapping solutions with the robot (three-dimensional representation) compared to just writing them out with paper and pencil (two-dimensional representation). While they did not report the results of the mental rotation assessments, their rationale for allowing students to use the robots to check the accuracy of their solutions and thus improve performance can be applied in other situations. Scratch provides a two-dimensional view of the sprites acting out the scripts on the screen whereas Cozmo physically acts out the code in three-dimensional reality. It is reasoned that this reification of the code will assist students in three-dimensional understanding and lead to higher scores in spatial ability for the Cozmo group.

Research also suggests the potential for robotics to develop spatial skills in K-12 students with findings of increased scores in spatial abilities for students participating in robotics interventions (Coxon, 2012; Julià & Antolí, 2016; Verner, 2004). Additionally, the research revealed issues with programming in virtual environments including reports that students can have problems giving directional commands when programming with objects (Fesakis et al., 2013; Kalelioglu, 2015). For example, students were searching for a button with a 'down' arrow when trying to move a sprite in the correct direction on the screen (Fesakis et al., 2013) which could be an issue in Scratch as well. Students can give commands to Cozmo and observe the results making the results concrete compared to the abstract visualization required in the virtual environment of Scratch.

Hypothesis 3

It is expected that competency beliefs will increase similarly for both training groups. This is the first time that the student participants in this study are being formally introduced to programming/coding in the school setting. Witherspoon et al. (2018) pointed out that providing opportunities for students to develop their identity as a programmer may be important for encouraging continued participation in computer science. However, exposing students to advanced content can have the undesirable side effect of reducing student confidence levels because they come to learn what competence in the domain actually involves (Witherspoon et al., 2018). Thus allowing students to experience "small wins" at each step of the programming process could raise students' perceived ability levels (Witherspoon et al., 2018). In the current study, it is reasoned that the use of visual programming blocks (Scratch Blocks) to introduce students to

programming in both the Scratch and Cozmo groups as opposed to using more complex text-based programming languages can decrease student cognitive load and allow students to achieve "small wins" thus increasing their perceived ability levels and competency beliefs.

This hypothesis is also based on empirical findings showing that both robotics (Hinton, 2018) and Scratch (Joshua Levi Weese, 2016) have been found to increase student self-efficacy. For instance, Weese (2016) found that two different Scratch interventions (video games vs. Mars rover) showed similarly strong improvements in self-efficacy. A qualitative study investigating the instructional use of a robotics educational curriculum on middle school students' attitudes toward and interests in STEM found that the robotics activities led to an increased interest and higher self-efficacy in STEM tasks (Hinton, 2018). Similarly, attendance in weekly robotics related courses or projects showed increases in self-efficacy in robotics (Kandlhofer & Steinbauer, 2016). Finally, a study that compared a short-term robotics intervention event (three-hour) to a longer (40-hour) robotics summer camp found that the longer intervention resulted in greater self-efficacy for youth ability to perform robotics tasks, and they had significantly greater confidence in their abilities than the students in the three-hour intervention although the students in the short intervention did have an increase from pre- to post in their self-efficacy with robotics (Nugent et al., 2015). The current study's intervention period (approximately 6-12 hours) exceeds the short-term three hour intervention period that resulted in increased self-efficacy, thus it is reasonable to expect increased self-efficacy for both groups in the current study.

Hypothesis 4

It is expected that the robotics training group will experience increased engagement compared to the computer-based training group. This hypothesis is based on observations and reports of students and teachers participating in several different empirical studies (Atmatzidou & Demetriadis, 2016; Phetsrikran et al., 2017; Pugnali & Sullivan, 2017; Witherspoon et al., 2017, 2018). A virtual robotics curriculum was tested with students in different contexts. When students used the curriculum with virtual robotics (simulated robots online), there was a decrease in interest scores for all groups from the beginning to the end of the intervention (Witherspoon et al., 2018), but in a second study using the same virtual robotics curriculum, teachers who used physical robots (in lieu of the online simulated robots) found that the use of physical robots improved student engagement and motivation to participate in class, and they attributed increased learning gains to the additional practice of programming skills in a similar context using physical robots (Witherspoon et al., 2017). Students using LEGO Mindstorms robotics kits reported that working with robots not only helped them develop deeper understanding of programming, but also kept them interested and motivated to keep working on programming (Atmatzidou & Demetriadis, 2016). Students using a researcher developed robot paired with an app on the iPad reported an interest in the robots and enjoyment writing programs to control the robots (Phetsrikran et al., 2017). And in a study where young children were offered a choice between using ScratchJr and a tangible robotics kit, the sessions using the robotics kits reached capacity a month

before the start date while none of the ScratchJr sessions reached capacity (Pugnali & Sullivan, 2017).

A recent analysis concluded that robots with physical presence were considered to be essential applications for educational robots for two reasons:

Studies have shown that the physical robot or embodied entity can elicit users' social behavior (Astrid et al., 2010), and they are perceived to be more engaging, enjoyable, sociable, and informative (Fasola & Mataric, 2013; Paauwe et al., 2015), and they can also lead to better user performance (Li, 2015). (Cheng et al., 2018, p. 400)

Additional findings support this notion. People rated a physical robot in the same room as more watchful and enjoyable than a simulated robot on a computer and a real robot that could be seen through teleconferencing (Wainer, Feil-Seifer, David, Shell, & Mataric, 2006). It was also discovered that children empathized significantly more with an embodied robot than a computer simulated robot after administering electric shocks to each type (both robots displayed colored bruises to indicate pain after being shocked) (Kwak et al., 2013).

In a recent study, Ozobots were found to elicit high motivation (Merino-Armero et al., 2018). Ozobots have a simple dome appearance (see Figure 3) and the only feature that they have that is meant to make them more social includes sounds meant to mimic human emotion. Cozmo is an autonomous emotional-educational robot with additional features that encourage interaction including artificial intelligence, animation and

emotions, facial recognition, and augmented reality capabilities made possible through computer vision. This high level of interactivity lends support to the hypothesis that Cozmo will be more engaging for students than working with the two-dimensional Scratch characters on a computer screen.

Significance of Study

This research contributes to an emerging body of literature on educational robotics in the teaching and learning of programming (Barker et al., 2010; Benitti & Barreto, 2012; Rogers & Portsmore, 2004; Williams et al., 2007) including spatial ability (Coxon, 2012), sequencing (Kazakoff & Bers, 2012; Sullivan & Bers, 2016; Sullivan et al., 2013), and computational thinking (Sullivan & Bers, 2016). Research in this area is particularly limited in formal school settings. While we know that robotics promotes learning in STEM and computer science, guidelines for integrating robotics into a curriculum have not been extensively researched yet.

The novelty of the use of educational-emotional robots in schools and their potential to appeal to both males and females and more generally to children with low interest in STEM is one that merits further investigation. The educational potential of the new emotional-educational robot Cozmo is already being reported (Giles, 2017; Gorman & Ackerman, 2017; C. Hermans, 2017; Kaiser, 2018; Kennington & Plane, 2017), but there is a lack of empirical evidence of its use as a learning tool in K-12 public schools. Research is needed to investigate the impact that Cozmo may have on student learning and motivation.

At a practical level, the current study may assist teachers and instructional designers with the design of programming activities for students. It can also directly benefit students through the development or enhancement of programming skills, competency, and engagement. Better understanding of the effects of coding interventions can also contribute to a better understanding of best practices and serve to better inform local, state, and/or national policies in regard to programming and STEM education.

This study empirically investigates the use of an emerging technology in a public school that provides equal access to all students. Initiatives that take place outside of school with select groups of students or intentionally manipulated in a laboratory setting thwart the goal of increasing the number and diversity of individuals pursuing programming and other STEM fields. Conducting the study in a natural setting in a public school is an important step toward attaining a clearer and more accurate picture of the effects that these tools and training may have.

CHAPTER III

METHODOLOGY

Introduction

The purpose of this chapter is to introduce the research methodology for this quasi-experimental quantitative study regarding the effects of coding activities supported by the artificially intelligent, animated emotive robot Cozmo on middle school students' computational thinking, spatial skills, competency beliefs, and engagement compared to the more traditional computer-based program of Scratch. The research plan, including the methodology, study participants, curriculum description, instructional methods, instruments, analysis method, and procedures are included in this chapter.

Research Question

Is there a difference in computational thinking performance, spatial abilities, competency beliefs, and engagement among middle school students who are taught coding using a traditional Scratch approach versus students who are taught using the same coding activities with Cozmo, an emotional-educational robot?

Hypotheses

Hypothesis 1: Cozmo is expected to produce greater gains in computational thinking.

Hypothesis 2: Cozmo is expected to produce greater learning gains in spatial skills.

105

Hypothesis 3: On average, both the Cozmo and Scratch training groups are expected to experience an increase in competency beliefs.

Hypothesis 4: Cozmo is expected to produce greater student engagement.

Participants

All seventh grade students enrolled in a course called "Technology" at a public middle school in the Midwest received one of the two training interventions. The regular classroom teacher assigned one class to each of the two interventions. The class that was assigned to the Scratch intervention contained 29 students and the class that was assigned to the Cozmo intervention contained 30 students. Table 1 contains demographic information for each full class.

Table 1

Student Demographics: All students

Student Characteristic	Scratch	Cozmo
Gender	Female = 19, Male = 10	Female = 15, Male = 15
Age	M = 12.61, SD = .094	M = 12.38, SD = .126
ESL	7	2
Gifted	3	1
Special Needs (IEP)	2	5

Note. ESL = English as a Second Language. IEP = Individualized Education Plan.

While all students in both classes received all of the training interventions, data are only reported for those students who returned signed Parent Consent and Student Assent forms. This resulted in 21 student participants in the Scratch intervention, and 22 student participants in the Cozmo intervention. Demographic information for the study participants who consented to have their data used in the study can be found in Table 2.

Table 2

Student Demographics: Study participants for which data is reported

Student Characteristic	Scratch	Cozmo
Gender	Female = 17, Male = 4	Female = 14, Male = 8
Age	M = 12.62, SD = .109	M = 12.36, SD = .155
ESL	5	2
Gifted	2	1
Special Needs (IEP)	2	3

Note. ESL = English as a Second Language. IEP = Individualized Education Plan.

Curriculum Description

The researcher developed two versions of the curriculum unit for this study. Both versions shared the same content and instructional features, but differed in the code blocks used based on what was available in each of the two programs. The computer-based training group used the materials as designed for Scratch. The robotics-based training group used adjusted blocks to fit those available in Cozmo's Code Lab app. The unit was included as part of the Technology curriculum at the middle school during the

last few weeks of the semester length Technology course and used to introduce students to coding/programming.

The intervention covered the International Society for Technology in Education (ISTE) Computational Thinker standard for students (https://iste.org/standards/for-students): "Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions." It also covered Topic 3 of the Design and Technology Strand for the Ohio Learning Standards in Technology (http://education.ohio.gov/getattachment/Topics/Learning-in-Ohio/Technology/Ohio-s-2003-Academic-Content-Standards-in-Technolo/The-2017-Ohio-Learning-Standards-in-Technology.pdf.aspx). The standards from the 6-8 grade band that were addressed included: "Collaborate to solve a problem as an interdisciplinary team modeling different roles and functions" and "Evaluate the effectiveness of the group's collaboration during the engineering design process and the contribution of the varying roles."

Each curriculum unit included a series of eight lessons: One for the computer-based intervention (Scratch) and the other for the robotics-based intervention (Cozmo). Scratch and Cozmo's Code Lab app both utilize visual programming languages that are based on a collection of graphical "programming blocks" that can be snapped together to create programs. The ScratchEd research team at the Harvard Graduate School of Education developed the Creative Computing Course in order to promote creativity and computational thinking and it can be used with everyone in grades K-12 and beyond (Brennan et al., 2011). Originally developed for Scratch, the current study adapted one of

the course units in order to provide middle school students with artificially intelligent robotics coding activities. The Scratch training group used the lessons as designed, and the materials for the Cozmo group were modified to fit the blocks available in the Code Lab app. Unit 1 (Exploring) was used with both groups and is described as allowing students to:

> Get comfortable with the key computational concept of sequence through a series of activities that provide varying levels of structure – from a step-by-step tutorial, to a creative challenge using a limited number of blocks, to open-ended explorations. (Brennan et al., 2011, p. 4)

The main difference between the two units was that the blocks in the robotics-based activities were modified to fit the blocks available in the Cozmo Code Lab app. Every effort was made to find the same or closest equivalent block when possible. For example, the Scratch blocks *Go to, Say, Play sound until done,* and *Repeat* were replaced with *Drive, Say, Play sound,* and *Repeat* in the 10 Block Challenge lesson.

Additional efforts were made to test the adapted Cozmo materials before implementation in the current study. These included an informal pilot test during the 2017-2018 school year with one male and one female seventh grade student at the middle school where the study took place. Modifications were made to the materials based on student feedback and responses to ensure student understanding. Dr. Chia-Ling Kuo, Educational Technology professor at Kent State University, reviewed the debugging challenges written for Cozmo to ensure that they covered the same concepts as those

covered in the Scratch lessons. A planning sheet and rubric were also developed to provide guidance for students with the challenge project. While the content of the projects was open-ended, the rubric outlined cohesiveness elements (e.g., clarity, features, appeal, originality, and functionality) and types of blocks (e.g., looks/display, events, sound, motion, control, actions, animations) that students should have included in their challenge projects to ensure similarity between the Scratch and Cozmo training conditions.

Table 3 shows an overview of the technology curriculum unit including the target concept, objective, and length of each lesson. Full detailed lesson plans can be found in Appendix A.

Table 3

Technology curriculum unit overview

Lesson	Target Concept	Objective	Length
1	Pretests	Students complete the Computational Thinking Test as a pretest for the unit.	1 class period (45 minutes)
2	Coding introduction	Students are introduced to what coders do and learn basic coding with code.org	1 class period (45 minutes)
3	Introduction to tool	Students are introduced to their assigned tool (Scratch/Cozmo) and engage in an exploratory, hands-on experience with it.	1 class period (45 minutes)
4	Step-by-step tutorial	Students follow a step-by-step tutorial and experience building up a program by experimenting and iterating.	1 class period (45 minutes)

5	10 block challenge	Students create a project with the constraint of only being able to use 10 blocks.	1 class period (45 minutes)
6	Debugging	Students investigate the problem and find a solution to five debugging challenges.	1-2 class periods (45-90 minutes)
7	Challenge!	Students become familiar with a wider range of blocks and create an open-ended project.	2 class periods (90 minutes)
8	Share/ Feedback	Students share and evaluate Challenge projects, then use feedback to adjust projects.	1 class period (45 minutes)
9	Post-tests	Students complete the Computational Thinking Test as a post-test for the unit.	1 class period (45 minutes)

Instructional methods

The researcher developed two versions of the curriculum unit. Both versions shared the same content and instructional features, but differed in the code blocks used based on what was available in each of the two programs. The computer-based training group used the materials as designed for Scratch. The robotics-based training group used adjusted blocks to fit those available in Cozmo's Code Lab app.

Computer-based instructional method

This instructional method included the curriculum as designed and served as a comparison for the robotics-based instructional method. The computer-based instructional method presented students with lessons after which they created and ran code using their own personal Scratch accounts on the desktop computers available in the

computer lab where the technology course took place. Figure 11 shows the Scratch user interface. Students drag and snap together their code blocks into the panel on the left side of the screen and the code is executed in the panel on the right side of the screen.
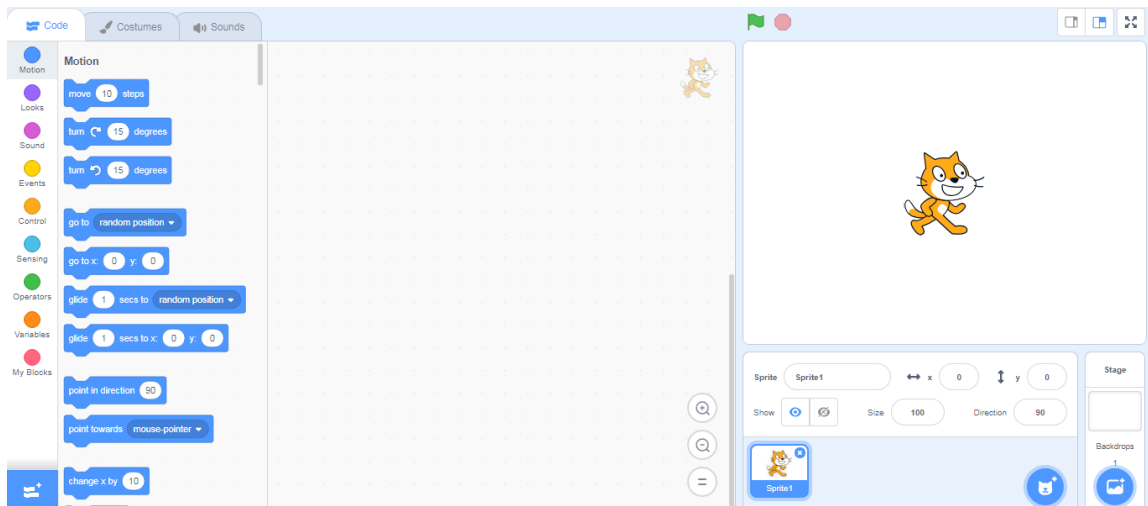

Figure 11. Scratch interface.

Robotics-based instructional method

The robotics-based instructional method adapted the blocks used in the computer-based instructional method to fit those available in the Cozmo Code Lab app. The robotics-based instructional method presented students with lessons after which they ran the code on the Cozmo Code Lab app on an iPad that was shared with at least one other student. After creating the code in the app, students tested the code on the Cozmo robot that communicates with the app via Wi-Fi signal. Figure 12 shows the Cozmo robot, charger, and three of Cozmo's Power Cubes. Figure 13 shows the Cozmo user interface (Constructor Mode) which is run through the Code Lab app on a mobile device.

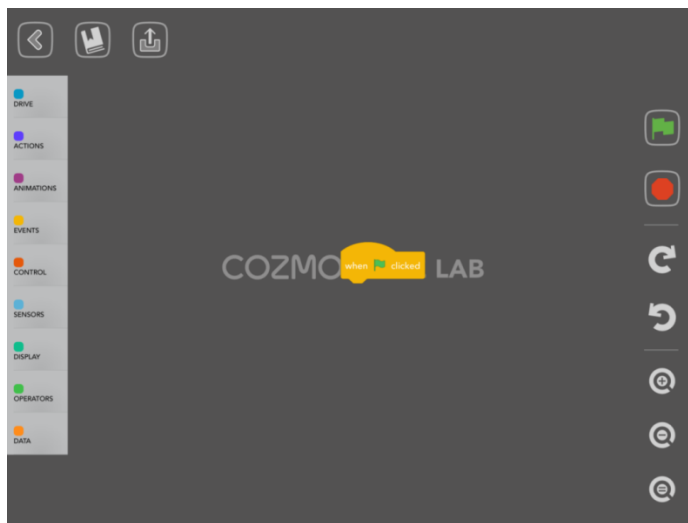*Figure 12.* Cozmo robot, charger, and three of Cozmo's Power Cubes.

Figure 13. Cozmo interface: Code Lab App (Constructor Mode).

Instruments

All four instruments were pilot tested in the spring of the 2017-2018 school year with one male seventh grade student and one female seventh grade student from the middle school where the study took place to ensure understanding and to estimate the length of time required for each assessment. Table 4 gives an overview of the instruments used in the study.

Table 4

Overview of study instruments

Dependent variable	Instrument	Time (min)	Implementation	Total Item Number	Score Range
Computational thinking	Computational Thinking Test (CTt) (Román-González et al., 2017)	45	Before and after intervention	28	0-28
Spatial skills	Mental Rotations Test (MRT) (Peters et al., 1995)	10	Before and after intervention	24	0-24
Competency beliefs	Technology Competency Beliefs Scale (Y.-F. Chen et al., 2017)	5	Before and after intervention	12	12-48
Student engagement	Engagement Scale (Chung et al., 2016)	5	After each lesson during intervention	8	8-32

Computational thinking

The Computational Thinking Test (CTt) used to measure computational thinking skills was developed by Román-González, Pérez-González, and Jiménez-Fernández (2017). The measured construct of computational thinking was operationally defined as "the ability to formulate and solve problems by relying on the fundamental concepts of computing, and using logic-syntax of programming languages: basic sequences, loops, iteration, conditionals, functions and variables" (Román-González, Pérez-González, & Jiménez-Fernández, 2017, p. 4).

The CTt was designed and intended for students between 12 and 14 years old (seventh and eighth grade) but can also be used in lower grades (fifth and sixth grade) and upper grades (ninth and tenth grade). The instrument consists of 28 multiple choice items with four answer options (only one correct) and estimated completion time is 45 minutes. One point is awarded for each correct response, thus the lowest possible score is zero and the highest possible score is 28.

The CTt was administered to a total sample of 1,251 fifth to tenth grade Spanish students from 24 different public and private schools of which 80% completed the test on a personal computer and 20% completed it on a tablet. Reliability as internal consistency of the CTt, measured by Cronbach's Alpha, was found to be α = 0.793 (Román-González et al., 2017, p. 9).

The items and response options in the CTt are based on popular websites for learning programming like Code.org (Román-González et al., 2017, p. 4). Each item is presented in 'The Maze' (23 items) or 'The Canvas' (five items) environment-interface and each item response alternative is presented with either visual arrows (eight items) or visual blocks (20 items). The authors granted permission for the researcher to use the test in the current study. See Figures 1-4 on pages 5-6 in Román-González, Pérez-González, and Jiménez-Fernández (2017) for examples of problems that can be found in the assessment. The authors intend for the CTt to be administered collectively and online via non-mobile or mobile electronic devices (Román-González et al., 2017, p. 4); however, the CTt was administered collectively in a hard copy paper/pencil version for the current study.

The 'computational concepts' covered in the assessment are aligned with some of the Computational Thinking (CT) framework (Brennan & Resnick, 2012) and with the Computer Science Teachers Association (CSTA) Computer Science Standards for seventh and eighth grade (CSTA, 2011). Each item addresses one or more of the following seven computational concepts, ordered in increasing difficulty:

- Basic directions and sequences (4 items)

- Loops – repeat times (4 items)

- Loops – repeat until (4 items)

- If – simple conditional (4 items)

- If/else – complex conditional (4 items)

- While conditional (4 items)

- Simple functions (4 items)

The CTt is also partially aligned with the 'computational practices' from the CT framework (Brennan & Resnick, 2012) depending on which of the following cognitive tasks is required for solving the item:

- Sequencing: the student must sequence, stating in an orderly manner, a set of commands (14 items)

- Completion: the student must complete an incomplete given set of commands (9 items)

- Debugging: the student must debug an incorrect given set of commands (5 items)

The experimental curriculum adapted for this study directly covers the computational concepts of basic directions, sequencing, and loops (repeat times) which aligns it with questions 1-8 on the CTt (see Table 5). Student scores for these questions will be calculated separately in the results section.

Table 5

CTt computational concepts covered by experimental curriculum

Item	Interface of item	Answer alternatives style	Computational concept addressed	Required task
1	The Maze	Visual arrows	Basic directions and sequences	Sequencing
2	The Maze	Visual arrows	Basic directions and sequences	Completion
3	The Maze	Visual blocks	Basic directions and sequences	Debugging
4	The Canvas	Visual blocks	Basic directions and sequences	Sequencing
5	The Maze	Visual arrows	Loops - Repeat times	Sequencing
6	The Maze	Visual arrows	Loops - Repeat times	Completion
7	The Canvas	Visual blocks	Loops - Repeat times	Debugging
8	The Maze	Visual blocks	Loops – Repeat times	Sequencing

The remaining questions on the CTt cover concepts that students may have discovered through their explorations with and completion of the final project of the curriculum.

Spatial skills

The Revised Vandenberg & Kuse Mental Rotations Test (MRT) was used to measure spatial skills (Peters et al., 1995). The reliability of the test is satisfactory. In a sample of 3,268 adults and adolescents age 14 or older, the Kuder-Richardson 20 (internal consistency reliability) was .88 and in a similar sample of 336 subjects, the test-retest correlation was .83 after an interval of a year or more and .70 for the same interval in an age corrected sample of 456 (Vandenburg & Kuse, 1978).

This paper/pencil test contains 24 items in which two-dimensional drawings of three-dimensional geometrical figures are to be compared. Each problem has a target figure shown on the left and four stimulus figures on the right. Two of these stimulus figures are rotated versions of the target figure, and two of the stimulus figures cannot be matched to the target figure. To score the instrument, one and only one point is given if both of the stimulus figures match the target figure and are identified correctly. No credit is given for a single correct answer. The maximum score attainable is 24 and the minimum score attainable is 0.

The test was introduced in a "serious and moderately personal manner" (Peters et al., 1995, p. 39) and then test administration procedures were followed as specified by Peters et al. (1995). Verbal instructions were given for problem set number 1 and then up

to five minutes was given for problem set number 2 (i.e., the three problems on page 2).
After these practice questions, the test began. Students were given four minutes to
complete pages 3 and 4, there was a short break, and then students were given four
minutes to complete pages 5-6.

The researcher was granted permission to use the test in the current study. See
Figure 1 on page 42 in Peters et al. (1995) for an example of a problem that can be found
in the assessment.

Competency beliefs

Competency Beliefs were measured using an adapted version of Activation Lab's
Competency Beliefs in STEM Scale version 1.0 (Y.-F. Chen et al., 2017). This
instrument is a 12-item survey with Likert scale response options from 1 to 4. The
maximum score attainable is 48 and the minimum score attainable is 12. The survey was
designed for 10-14 year-old respondents to measure an individual's STEM Competency
Beliefs at the time the survey responses are collected. Competency beliefs were defined
as "people's judgments of their capabilities to organize and execute courses of action
required to attain designated types of performances" (Bandura, 1986). The reliability of
the STEM Competency Beliefs Scale is acceptable (Cronbach's alpha = 0.83; polychoric
alpha = 0.87) based on analyses of a total sample of 205 middle school youth (Chen et al.,
2017, p. 2).

The prompts and response options and coding for the original version of the survey can be seen in Figure 14 and can be found online at http://activationlab.org/wp-content/uploads/2018/03/CompetencyBeliefs_STEM-Report_20170403.pdf.

Survey Questions
The STEM Competency Beliefs scale has 12 items, with response options from 1 to 4.

Item ID	Prompt	Response Options and Coding
CB1	I can do the math problems I get in class	4=All of the time; 3=Most of the time; 2=Half the time; 1=Rarely
CB2	I can understand scientific information on websites for kids my age	4=All websites; 3=Most websites; 2=A few websites; 1=None of them
CB3	If I did my own project in an after school science club, the project would be…	4=Excellent; 3=Good; 2=Okay; 1=Poor
CB4	I am the technology expert in my house	4=All of the time; 3=Most of the time; 2=Half the time; 1=Rarely
CB5	I can understand the science in books for adults	4=All of the time; 3=Most of the time; 2=Some of the time; 1=A little of the time
CB6	I think I am very good at: Figuring out how to fix things that don't work.	4=YES!; 3=yes; 2=no; 1=NO!
CB7	I think I am very good at: Giving evidence when I tell my opinion.	4=YES!; 3=yes; 2=no; 1=NO!
CB8	I think I am very good at: Explaining my solutions to math problems.	4=YES!; 3=yes; 2=no; 1=NO!
CB9	I think I am very good at: Solving problems.	4=YES!; 3=yes; 2=no; 1=NO!
CB10	I think I am very good at: Coming up with my own science investigations.	4=YES!; 3=yes; 2=no; 1=NO!
CB11	I think I am very good at: Coming up with new ways to solve technical problems.	4=YES!; 3=yes; 2=no; 1=NO!
CB12	I think I am very good at: Coming up with new ideas when working on projects.	4=YES!; 3=yes; 2=no; 1=NO!

Figure 14. Original version of the STEM Competency Beliefs Survey.

The researcher changed some of the wording of the original survey to focus on technology and coding in order to better fit the needs of the current study, and the survey was renamed Technology Competency Beliefs Scale. A pilot study was conducted by the researcher with one male and one female seventh grade student. It took students less than 5 minutes to complete the survey and students confirmed that the survey was

understandable as written. The adapted prompts and answer response options and coding

of the survey can be found in Figure 15.

Item	Prompt	Response Options and Coding
1	I can do coding projects I get in class	4=All of the time; 3=Most of the time; 2=Half the time; 1=Rarely
2	I can understand coding for kids my age	4=All websites; 3=Most websites; 2=A few websites; 1=None of them
3	If I did my own project in an after school coding club, the project would be…	4=Excellent; 3=Good; 2=Okay; 1=Poor
4	I am the technology expert in my house	4=All of the time; 3=Most of the time; 2=Half the time; 1=Rarely
5	I can understand computer programming languages for adults	4=All of the time; 3=Most of the time; 2=Some of the time; 1=A little of the time
6	I think I am very good at: Figuring out how to fix things that don't work:	4=YES!; 3= yes; 2=no; 1=NO!
7	I think I am very good at: Giving evidence when I tell my opinion.	4=YES!; 3= yes; 2=no; 1=NO!
8	I think I am very good at: Explaining my solutions to coding problems.	4=YES!; 3= yes; 2=no; 1=NO!
9	I think I am very good at: Solving coding problems.	4=YES!; 3= yes; 2=no; 1=NO!
10	I think I am very good at: Coming up with my own coding projects.	4=YES!; 3= yes; 2=no; 1=NO!
11	I think I am very good at: Coming up with new ways to solve technical problems.	4=YES!; 3= yes; 2=no; 1=NO!
12	I think I am very good at: Coming up with new ideas when working on projects.	4=YES!; 3= yes; 2=no; 1=NO!

Figure 15. Technology Competency Beliefs Scale.

Student engagement

Student engagement was measured using Activation Lab's Engagement in Science Learning Activities survey version 3.2 (Chung et al., 2016). The title was shortened to simply "Engagement Scale" for the current study to avoid student confusion. Previous research of a sample of 2,600 youth from sixth and eighth grade science classrooms indicated that both the raw (Cronbach's) and polychoric alpha coefficients were found to be acceptable (.80 and .85, respectively). The survey contains eight 4-point Likert scale items and was written for use with 10-14 year-old respondents immediately after an activity. It measures a respondent's self-reported cognitive, behavioral, and affective engagement during the learning opportunity and is intended for formative feedback and/or research purposes. The authors conceptualized engagement (a person's focus, participation, and persistence on a task) as having three dimensions: behavioral (related to task completion or off task); cognitive (thought processes and attention directed toward meaningful information processing involved in task completion); and affective (emotions during task completion are positive and high arousal or negative and low arousal).

This provides three scores; an overall engagement score, a score for affective engagement, and a score for behavioral/cognitive engagement. Pragmatically, scores can be produced from simple averages of all items (all of which are based on a 4-point Likert scale, with reverse coding for four of the items) to give an overall engagement score; or for the sub-parts, the sum of items for affective engagement or behavioral/cognitive

engagement. Simple averages appear to have stronger predictive validity than factor

scores (Chung et al., 2016, paragraph 4).

In the current study, an overall score for the whole scale was calculated. The

maximum score attainable is 32 and the minimum score attainable is 8. Students took

less than five minutes to complete the survey. A copy of the prompts, sub-factors, and

response and options coding can be seen in Figure 16 and can be found online at

http://activationlab.org/wp-content/uploads/2018/03/Engagement-Report-3.2-

20160803.pdf.

The Engagement Scale was administered to students at the conclusion of each

day's lesson. The researcher used the engagement scores as a type of formative feedback

to indicate the engagement qualities of each lesson. In addition, all engagement scores

for each participant were averaged and then a linear mixed model was conducted to

determine which intervention was more engaging overall.

The Instrument
Engagement in Science

Item ID Number	Prompt	Sub-factor	Response Options and Coding
E01*	During this activity: I felt bored.	Affect	1=YES! 2=yes 3=no 4=NO!
E02	During this activity: I felt happy.	Affect	4=YES! 3=yes 2=no 1=NO!
E03	During this activity: I felt excited.	Affect	4=YES! 3=yes 2=no 1=NO!
E04*	During this activity: I was daydreaming a lot.	Cognitive	1=YES! 2=yes 3=no 4=NO!
E05	During this activity: I was focused on the things we were learning most of the time.	Cognitive	4=YES! 3=yes 2=no 1=NO!
E06	During this activity: Time went by quickly.	Behavior	4=YES! 3=yes 2=no 1=NO!
E07*	During this activity: I was busy doing other tasks.	Behavior	1=YES! 2=yes 3=no 4=NO!
E08*	During this activity: I talked to others about stuff not related to what we were learning.	Behavior	1=YES! 2=yes 3=no 4=NO!

*Item is reverse-coded.

Figure 16. Student Engagement Scale.

Procedures

Two intact seventh grade classes participated in the study during their regularly scheduled Technology course. The regular classroom teacher chose which class was assigned to each of the two training groups. The Student Demographics Survey was attached to the Parent Consent form for completion by students who consented to participate in the study. All of the students in both classes received the training, but data are reported only for students who returned signed Parent Consent and Student Assent forms. Students completed the Technology Competency Beliefs survey and the Mental

Rotations Test when signed Parent Consent and Student Assent forms and Student

Demographics Surveys were collected. Figure 17 shows an overview of the procedures
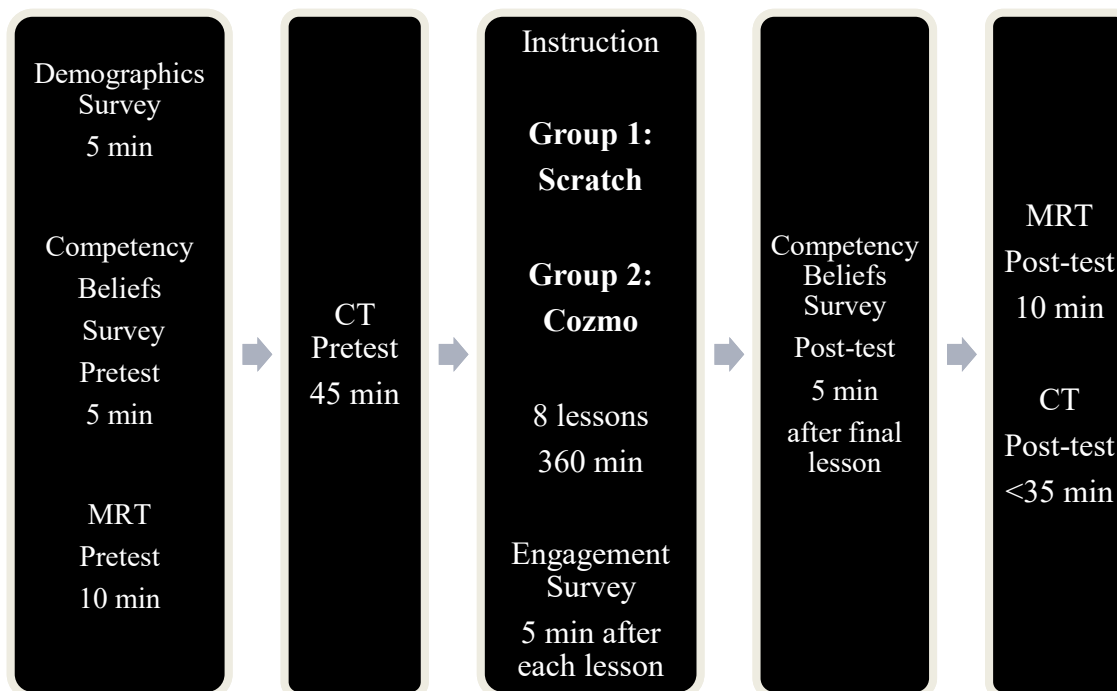
for the study.

| Demographics Survey 5 min

Competency Beliefs Survey Pretest 5 min

MRT Pretest 10 min | CT Pretest 45 min | Instruction

Group 1: Scratch

Group 2: Cozmo

8 lessons 360 min

Engagement Survey 5 min after each lesson | Competency Beliefs Survey Post-test 5 min after final lesson | MRT Post-test 10 min

CT Post-test <35 min |

Figure 17. Overview of study procedures.

Students completed the Demographics Survey, Competency Beliefs Survey,

Mental Rotations Test, and Computational Thinking Pretest before winter break in

December 2018. Students began learning computer programming lessons after returning

from winter break in January 2019. One class used a program on the computer called

Scratch and the other class used an app on the iPad called *Code Lab* to operate a robot

named *Cozmo*. The researcher conducted all lessons.

During the first class period (pretesting), final class period (post-testing), and introduction to coding (class period 2), students in both classes remained in their regularly assigned seats in the computer lab where the technology class took place. During the remainder of the instructional intervention (class periods 3-9), the Scratch group worked in the same computer lab while the Cozmo group worked in the high school library.

Students in the Scratch intervention were permitted to choose their own seats in the computer lab of their regularly scheduled class where seats were arranged in rows that ran the perimeter of the room. There was no assignment of partners and/or groups, but students were permitted and encouraged to ask each other questions and for help as needed. While students were seated next to or between friends and could collaborate as desired, each student completed all work individually at the computer of the self-selected seat.

Students in the Cozmo intervention were permitted to choose a partner and the table at which they would like to work in the high school library. Students asked if they could work in larger groups. Consent was granted, with a cap of four students per group. Most groups remained stable throughout the intervention, but students were permitted to change groups if desired. The engagement surveys revealed the lowest possible scores for two students in the same group during class period seven. The researcher checked in with the students the following day to make sure equipment was in working order, and one student explained that he was not getting along well with one of his partners because his partner was "hogging the iPad and won't let me do anything." The researcher

encouraged the student to find a new partner or group, and the student left the group of three to form a group of two with another student with whom he worked better.

Four of the Cozmo groups (three pairs and one group of three) remained stable throughout all of the lessons and the project. The other groups contained three to four students and were fluid with students changing groups from one lesson to the next until they found a group with which they worked better. The researcher noted that in groups of four, there were students who did not get to spend as much time using the iPad, so a cap of three students was put in place for the final project. This resulted in all students pairing with one other student with the exception of three groups with three students.

Class period 1 – Lesson 1

The Computational Thinking Test was administered in December 2018 as a pretest (45 minutes). This test did not affect students' grades in any way. Its purpose was to show how much students had already learned about programming.

Class period 2 – Lesson 2

Both training groups received an identical introduction to coding using Code.org on the first day after winter break in January 2019. After some initial instruction, students logged in to their own Code.org accounts and began solving coding puzzles. They completed an engagement survey at the end of the class period.

Class period 3 – Lesson 3

In lesson 3, both groups were introduced to the assigned technology and then engaged in an exploratory, hands-on experience with it. Each student in the Scratch group worked at a desktop computer in the computer lab where the technology course took place. Each student logged in to an individual Scratch account and completed the exploratory activities using the desktop computer.

Students in the Cozmo group worked in self-selected small groups of two, three, or four students. Each group of students shared an iPad on which to run the Code Lab app to program the Cozmo robot. Lessons three through eight for the robotics-based training group took place in the high school library because the iPads being borrowed for the study belonged to the high school and were connected to the high school's Wi-Fi. The middle school and high school were on the same campus and the buildings were physically connected.

Each pair of students also received a Cozmo robot, charger, and three of Cozmo's Power Cubes and some resource pages to explain the interface and blocks available in the Cozmo Code Lab app. After connecting the iPad to the robot's Wi-Fi network, students worked in small groups with their given materials to complete the exploratory activities. Students answered some reflection questions and completed an engagement survey at the end of the class period. Feedback from reflection questions at the end of class periods three through eight served as formative feedback for the researcher to improve lessons.

Class period 4– Lesson 4

This lesson led students through step-by-step tutorials of what to do. Students followed tutorial instructions to create a dancing cat (Scratch) or a dancing robot (Cozmo). Students were encouraged to evaluate and share their projects with other students/groups. Finally, students answered some reflection questions and completed an engagement survey at the end of the class period.

Class period 5– Lesson 5

This lesson gave students a minimum number of coding blocks with which to work. Students created a project with the constraint of only being able to use 10 blocks. Students were given the list of 10 blocks that could be used, reminded to use each block at least once in their project, and encouraged to experiment. Students were encouraged to evaluate and share their projects with other students/groups. Finally, students answered some reflection questions and completed an engagement survey at the end of the class period.

Class period 6 – Lesson 6

This lesson helped students figure out how to debug (fix the mistakes) in code so that it would run correctly. Students were given five debugging challenges for which they investigated the buggy program, tinkered with the problematic code, and tested possible solutions. Finally, students answered some reflection questions and completed an engagement survey at the end of the class period.

A catch-up class period was added after the initial debugging lesson for the Cozmo group for several reasons. More students were absent and missed the instructional lessons with their assigned tool in the Cozmo group ($n = 9$) compared to the Scratch group ($n = 4$). Additionally, the Cozmo group was using older iPads in the high school library. The library was arranged into four areas: a computer lab, a second open computer lab area equipped with Chromeboxes, a third lounging area where students could use their cell phones, and an area equipped with tables and chairs. The Cozmo group used the area with tables and chairs. Each group of students worked at one of the tables.

During Lesson Three (Introduction to Tool), the Cozmo group was the only group using the library and there were no technical issues at all. However; during Lessons Four, Five, and Six (all of the formal instructional lessons with Cozmo), there were high school classes that came in to use the Chromebox lab, there were many students in the lounging section using their cell phones, and there were students using computers in the computer lab. The age of the iPads coupled with the significant reduction in bandwidth due to the high number of devices connected to the Wi-Fi in the library during this time caused the Cozmo app to crash repeatedly for most of the groups during all three days of instruction.

Class periods 7-8 – Lesson 7

This lesson put everything together and allowed students to create and share their own coding projects. Students worked on creating and testing their own open-ended

projects during these class periods. While the content of the project was open-ended, students were provided guidance throughout the process of completing their projects. First, students received a planning sheet to help them get started and ensure that their project was cohesive. The planning sheet addressed the elements of clarity, features, originality, and functionality. Next, students received a rubric to guide them throughout the design process. The rubric was developed to provide guidance for students with the project cohesiveness elements (e.g., clarity, features, appeal, originality, and functionality) and types of blocks (e.g., looks/display, events, sound, motion, control, actions, animations) to include in their challenge projects (see Appendix B). These efforts helped ensure similarity between the Scratch and Cozmo training conditions. Students answered some reflection questions and completed an engagement survey at the end of each class period.

Class period 9 – Lesson 8

This lesson gave students the opportunity to share and receive peer feedback on their open-ended projects. Students tested a classmate(s) project and used a critique paper to evaluate it. Original plans allocated two days for this lesson so that students could share and receive feedback from multiple groups, but a reduction in the number of available instructional days beyond the researcher's control (e.g., two good behavior award days and a snow day) resulted in students only being able to give and receive feedback from one other student/group. Students then used the feedback from their classmates to make improvements to their projects. At the end of the period, students completed an engagement survey and the Competency Beliefs Survey.

Class period 10 – Lesson 9

The Mental Rotations Test was given and the Computational Thinking Test was administered as a post-test. Its purpose was to see if there was a difference in score from the pretest after completing the curriculum unit. It is important to note that the circumstances of this lesson differed for the Scratch group compared to the Cozmo group due to circumstances beyond the researcher's control. The Scratch group had a full class period and was able to complete the Mental Rotations Test and the Computational Thinking Test without issue. The Cozmo group should also have had a full class period to complete the Mental Rotations Test and the Computational Thinking Test, but adverse weather conditions on the originally-scheduled day (which was one day later than the Scratch group due to the added catch-up day caused by technical difficulties for the Cozmo group) resulted in a district-wide snow day. This moved Lesson 9 for the Cozmo group to the last day of the semester. The regular classroom teacher had to give a state-required assessment on that day because she would not see her students again (Technology is a semester-long class). This resulted in the following structure of the class period: the timed Mental Rotations Test was given first, the Computational Thinking Test was given second, and students were told to complete their technology assessment for the regular teacher when they finished the Computational Thinking Test. This resulted in many students rushing through the Computational Thinking Test (for which they were not receiving a grade) in order to complete the required technology assessment (for which they could receive an exam grade that would affect their overall grade for the class).

Dependent Variables

The dependent variables include computational thinking, spatial skills, competency beliefs, and student engagement. Computational thinking was measured using the Computational Thinking Test developed by Román-González, Pérez-González, and Jiménez-Fernández (2017). Spatial skills were measured using a revised version of the Mental Rotations Test developed by Vandenburg and Kuse (1978). Competency Beliefs were measured using an adapted version of Activation Lab's Competency Beliefs in STEM survey version 1.0 (Y.-F. Chen et al., 2017). Student engagement was measured using Activation Lab's Engagement in Science Learning Activities survey version 3.2 (Chung et al., 2016).

CHAPTER IV

ANALYSIS OF THE FINDINGS

Introduction

This chapter examines the results for this quasi-experimental quantitative study regarding the effects of coding activities supported by the artificially intelligent, animated emotive robot Cozmo on middle school students' computational thinking, spatial skills, competency beliefs, and engagement compared to the more traditional computer-based program of Scratch. Repeated measures analysis of variance with a between-subjects factor of intervention (Scratch/Cozmo) and within-subjects factor of time (pre/post-test) was used to analyze the effects of the two different instructional interventions on student computational thinking, spatial skills, and competency beliefs. A linear mixed model was used to analyze the effects of the two different instructional interventions on student engagement. Twenty-one students in the Scratch intervention and 22 students in the Cozmo intervention consented to have their data used in the study.

Measures

The dependent variables in this study were computational thinking, spatial skills, competency beliefs, and engagement. Computational thinking was measured by the 28 question Computational Thinking Test (CTt) administered before and after the training intervention. Spatial skills were measured by the 24 question Mental Rotations Test (MRT) administered before and after the training intervention. Competency beliefs were

134

measured by the 12 question Technology Competency Beliefs Scale administered before and after the training intervention. Student engagement was measured by the eight question Engagement Scale administered after each lesson during the training intervention.

All four dependent variables were continuous: Computational Thinking (CTt total score), Spatial Skills (MRT total score), Competency Beliefs (Competency Beliefs Survey total score), and Engagement (Engagement Survey average score). The independent variable was categorical (Treatment: Scratch versus Cozmo).

Repeated measures ANOVA were computed to compare students' pre- and post-test data in computational thinking, mental rotation, and competency beliefs. Repeated measures ANOVA was also computed for all of the previously mentioned data without including scores for students with special education needs (IEP) and with English as a Second Language (ESL) because comprehension of instructions and questions on each assessment was an issue for these students. A linear mixed model was used to analyze the effects of the two different instructional interventions on student engagement. Table 6 provides a summary of means and standard deviations for pre- and post-test data for each dependent variable for all students. The first day of instruction (class period 2) is reported as the pretest for engagement and the last day of instruction (class period 9) is reported as the post-test for engagement. Table 7 provides a summary of means and standard deviations for pre- and post-test data for each dependent variable without ESL and IEP student data. Table 8 provides a summary of means and standard deviations for student engagement during each day of the study for the Scratch group. Table 9 provides

a summary of means and standard deviations for student engagement during each day of the study for the Cozmo group.

Table 6

Means and standard deviations (All students): CTt, MRT, Competency Beliefs, Engagement

Dependent Variable	Groups			
	*Scratch ($n = 20$)		Cozmo ($n = 22$)	
	Pre *M* (*SD*)	Post *M* (*SD*)	Pre *M* (*SD*)	Post *M* (*SD*)
CTt total score	16.65 (4.31)	17.30 (5.25)	16.27 (5.07)	16.82 (5.32)
MRT	7.55 (4.67)	10.90 (6.25)	6.09 (4.10)	10.91 (4.67)
Competency Beliefs	30.70 (6.66)	33.85 (7.61)	30.86 (6.62)	34.09 (6.57)
Engagement**	($n = 19$) 2.93 (.51)	($n = 19$) 2.97 (.50)	3.19 (.54)	3.26 (.45)

Note. *Scores are not reported for one student who was absent during pretesting.

**Engagement scores are reported for all students present during the first (pretest) and last (post-test) days of the intervention.

Table 7

Means and standard deviations (Minus ESL and IEP): CTt, MRT, Competency Beliefs,

Engagement

Dependent Variable	Groups			
	*Scratch ($n = 13$)		Cozmo ($n = 17$)	
	Pre *M* (*SD*)	Post *M* (*SD*)	Pre *M* (*SD*)	Post *M* (*SD*)
CTt total score	18.46 (3.53)	19.85 (4.47)	17.00 (5.45)	18.53 (4.11)
MRT	9.54 (3.93)	12.77 (6.07)	6.71 (4.36)	11.65 (4.94)
Competency Beliefs	32.00 (6.61)	36.23 (7.40)	30.71 (6.90)	35.53 (6.66)
Engagement**	($n = 13$) 2.89 (.38)	($n = 12$) 2.96 (.47)	($n = 17$) 3.18 (.59)	($n = 17$) 3.34 (.44)

Note. *Scores are not reported for one student who was absent during pretesting.

**Engagement scores are reported for all students present during the first (pretest) and

last (post-test) days of the intervention.

Table 8

Means and standard deviations: Scratch daily engagement

Time	Groups	
	All students	Minus ESL & IEP
	M (SD)	M (SD)
Class period 2	(n = 19) 2.93 (.51)	(n = 13) 2.89 (.38)
Class period 3	(n = 21) 3.11 (.49)	(n = 14) 3.04 (.49)
Class period 4	(n = 19) 3.03 (.44)	(n = 12) 3.00 (.41)
Class period 5	(n = 20) 3.05 (.40)	(n = 13) 3.07 (.40)
Class period 6	(n = 21) 3.01 (.46)	(n = 14) 2.93 (.46)
Class period 7	(n = 21) 2.98 (.51)	(n = 14) 2.96 (.47)
Class period 8	(n = 20) 3.06 (.47)	(n = 14) 3.02 (.46)
Class period 9	(n = 19) 2.97 (.50)	(n = 12) 2.96 (.47)

Table 9

Means and standard deviations: Cozmo daily engagement

Time	Groups	
	All students	Minus ESL & IEP
	M (*SD*)	*M* (*SD*)
Class period 2	(*n* = 22) 3.19 (.54)	(*n* = 17) 3.18 (.59)
Class period 3	(*n* = 20) 3.56 (.41)	(*n* = 15) 3.61 (.41)
Class period 4	(*n* = 21) 3.43 (.43)	(*n* = 16) 3.47 (.43)
Class period 5	(*n* = 22) 3.35 (.45)	(*n* = 17) 3.39 (.44)
Class period 6	(*n* = 19) 3.11 (.67)	(*n* = 15) 3.15 (.68)
Make-up period	(*n* = 19) 3.13 (.54)	(*n* = 17) 3.06 (.58)
Class period 7	(*n* = 21) 2.98 (.69)	(*n* = 14) 3.19 (.58)
Class period 8	(*n* = 22) 3.26 (.64)	(*n* = 17) 3.30 (.67)
Class period 9	(*n* = 22) 3.26 (.45)	(*n* = 17) 3.34 (.44)

Student Background Information

Additional demographic information was collected from study participants using a survey modeled after Weese, Feldhausen, and Bean (2016). Table 10 reports the availability of different resources in students' homes. Analysis of variance revealed no significant differences in availability of resources between the Scratch and Cozmo groups (see Table 11). Table 12 reports information concerning different types of learning assistance available to students at home. Analysis of variance revealed no significant differences in types of learning assistance available at home between the Scratch and Cozmo groups (see Table 13). Table 14 shows students' programming experience. An analysis of student background questionnaires revealed that none of the participants had previous formal training in coding.

Table 10

Resources available to students at home (Mean scores)

Resource	Scratch ($n = 21$)	Cozmo ($n = 22$)
	M (*SD*)	*M* (*SD*)
Calculator	3.52 (.814)	3.45 (.800)
Computer (Not video game systems)	3.05 (1.07)	3.23 (.813)
Internet access	3.43 (.870)	3.55 (.510)
Dictionary	2.67 (1.11)	2.45 (1.26)

Study/homework area	3.43 (.978)	3.23 (.973)
E-reader (e.g., iPad, Kindle, Nexus)	2.57 (1.29)	2.41 (1.22)

Note. Always = 4, Most of the time = 3, Rarely = 2, Never = 1.

Table 11

ANOVA: Resources available to students at home

Variable	*df*	*F*	*p*	Cohen's *d*
Calculator	42	.08	.78	-0.09
Computer	42	.39	.54	0.19
Internet access	42	.29	.59	0.17
Dictionary	42	.34	.56	-0.19
Study/homework area	42	.46	.50	-0.21
E-reader	42	.18	.67	-0.13

* *p* < .05. ** *p* < .01.

Table 12

Learning assistance available to students at home (Mean scores)

Type of learning assistance	Groups	
	Scratch (*n* = 21)	Cozmo (*n* = 22)
	M (*SD*)	*M* (*SD*)
Student learning is important to someone.	3.24 (1.04)	3.55 (.596)
Someone can help with homework if needed.	3.10 (.889)	3.14 (.774)

Someone is interested in teaching the student new things.	3.00 (.894)	2.77 (.973)
Someone takes the student places to learn new things.	2.90 (.831)	2.64 (1.05)
Someone makes sure the student finishes homework daily.	3.10 (1.18)	3.41 (.854)
Someone at home helps with coding and computer programming.	1.00 (.000)	1.05 (.213)

Note. YES! = 4, yes = 3, no = 2, NO! = 1.

Table 13

ANOVA: Learning assistance available to students at home

Variable	*df*	*F*	*p*	Cohen's *d*
Student learning is important to someone.	42	1.42	.24	0.37
Someone can help with homework if needed.	42	.03	.87	0.05
Someone is interested in teaching the student new things.	42	.63	.43	-0.25
Someone takes the student places to learn new things.	42	.86	.36	-0.27
Someone makes sure the student finishes homework daily.	42	1.01	.32	0.30
Someone at home helps with coding and computer programming.	42	.95	.34	0.33

* $p < .05$. ** $p < .01$.

Table 14

Student programming experience

Type of experience	Number of students	
	Scratch	Cozmo
At school		
No previous experience	3	4
Hour of Code	18	18
Scratch	0	0
Blockly	0	0
Text-based	0	0
At a program, workshop, or contest outside of school		
No previous experience	17	18
Hour of Code	4	3
Scratch	0	1
Blockly	0	0
Text-based	0	0
At home		
No previous experience	13	16
Hour of Code	8	4
Scratch	0	1
Blockly	0	1
Text-based	0	0

Assumptions

A Shapiro-Wilk test was performed to test the assumption of normality. Table 15 shows that scores for all dependent variables did not deviate significantly from normal with the exception of the MRT pre- and post-test, and the engagement post-test (final day of engagement). Table 16 shows that scores for all dependent variables did not deviate significantly from normal with the exception of the MRT pretest. Levene's test was performed to test the assumption of homogeneity of variance. Table 17 shows that the variances were equal for all dependent variables for all students. Table 18 shows that the variances were equal for all dependent variables without ESL and IEP student data.

Table 15

Assumption of normality: Shapiro-Wilk test results (All students)

Dependent Variable	*df*		*F*		*p*	
	Pre	Post	Pre	Post	Pre	Post
CTt total score	42	43	.965	.975	.23	.47
MRT	42	43	.932	.939	.02*	.02*
Competency Beliefs	42	43	.969	.978	.31	.56
Engagement	41	41	.952	.937	.08	.03*

$* p < .05. ** p < .01.$

Table 16

Assumption of normality: Shapiro-Wilk test results (Minus ESL and IEP)

Dependent Variable	df		F		p	
	Pre	Post	Pre	Post	Pre	Post
CTt total score	30	31	.929	.968	.05	.46
MRT	30	31	.919	.933	.03*	.05
Competency Beliefs	30	31	.964	.973	.39	.60
Engagement	30	29	.950	.936	.17	.08

* $p < .05$. ** $p < .01$.

Table 17

Assumption of homogeneity of variance: Levene's test results (All students)

Dependent Variable	df		F		p	
	Pre	Post	Pre	Post	Pre	Post
CTt total score	40	40	1.150	0.035	.29	.85
MRT	40	40	0.258	4.417	.61	.05
Competency Beliefs	40	40	0.166	0.577	.69	.45
Engagement	37	37	0.007	0.021	.93	.89

* $p < .05$. ** $p < .01$.

Table 18

Assumption of homogeneity of variance: Levene's test results (Minus ESL and IEP)

Dependent Variable	*df*		*F*		*p*	
	Pre	Post	Pre	Post	Pre	Post
CTt total score	28	28	4.168	0.090	.05	.77
MRT	28	28	0.192	1.313	.67	.26
Competency Beliefs	28	28	0.002	0.089	.96	.77
Engagement	26	26	0.630	0.004	.44	.95

* *p* < .05. ** *p* < .01.

Analysis of variance (ANOVA) revealed non-significant ($p > 0.5$) differences between the Scratch and Cozmo intervention groups for all variables measured at the outset of the study for all students as well as for students without English as a Second Language (ESL) or special education needs (IEP). Engagement scores during class period 2 served as the pretest for engagement because instruction in Code.org on the computer was identical for both groups. Table 19 presents ANOVA results and effect sizes for all students, and Table 20 presents ANOVA results and effect sizes not including ESL and IEP student data.

Table 19

ANOVA: Pretests (All students)

Variable	df	F	p	Cohen's d
CTt	41	0.07	.80	-0.08
MRT	41	1.16	.29	-0.33
Competency Beliefs	41	0.01	.94	0.02
Engagement: Class period 2	40	2.47	.12	0.41

* $p < .05$. ** $p < .01$.

Table 20

ANOVA: Pretests (Minus ESL and IEP)

Variable	df	F	p	Cohen's d
CTt	29	0.71	.41	-0.32
MRT	29	3.39	.08	-0.68
Competency Beliefs	29	0.27	.61	-0.19
Engagement: Class period 2	29	2.40	.13	0.48

* $p < .05$. ** $p < .01$.

Results

A repeated measures analysis of variance with a between-subjects factor of intervention (Scratch/Cozmo) and within-subjects factor of time (pre/post-test) was performed to test the intervention's effects on computational thinking, spatial skills, and competency beliefs. This analysis provided information about the effect of the intervention (between the Scratch and Cozmo groups), pre-post time effects (within the Scratch group and within the Cozmo group), and the interaction between them (i.e., whether one intervention improved the dependent variable more than the other intervention). A linear mixed model was used to analyze the effects of the two different instructional interventions on student engagement. Intervention (Scratch versus Cozmo), class period, and the interaction of intervention by class period were set as the fixed factors, and class period was set as the repeated factor with compound symmetry covariance structure. This analysis provided information about whether students found Scratch or Cozmo more engaging, which class periods students found more engaging, and whether engagement varied between the Scratch and Cozmo groups during any of the class periods.

Computational Thinking

A repeated measures analysis of variance with a between-subjects factor (Cozmo versus Scratch) and within-subjects factor (CTt pre- and post-test) was performed to test the intervention's effects on computational thinking. On average, students' computational thinking knowledge was relatively low (All students: Scratch

approximately 59% correct, Cozmo approximately 58% correct; Not including ESL and IEP student data: Scratch approximately 66% correct, Cozmo approximately 61% correct) on the pretest.

CTt: Questions 1-28 (All students). There was a non-significant main effect of intervention, F (1, 40) = 0.090, p = .766, indicating insufficient evidence that computational thinking skills were different between the Scratch and Cozmo groups. There was a non-significant main effect of time, F (1, 40) = 1.012, p = .321, indicating that performance on the Computational Thinking Test was generally the same for the Scratch and Cozmo groups from pretest to post-test. There was a non-significant interaction between time and intervention, F (1, 40) = 0.008, p = .930. This effect indicates that both interventions had the same effect on computational thinking skills. Specifically, scores were similar between the Scratch (pretest, M = 16.65, SD = 4.31; post-test, M = 17.30, SD = 5.25) and Cozmo (pretest, M = 16.27, SD = 5.07; post-test, M = 16.82, SD = 5.32) groups (see Figure 18). There seems to be a very small effect size between the pre- and post-test computational thinking measures for Scratch (d =.14) and Cozmo (d = .11), as well as between the two learning conditions relative to the post-test score (d = -.09).
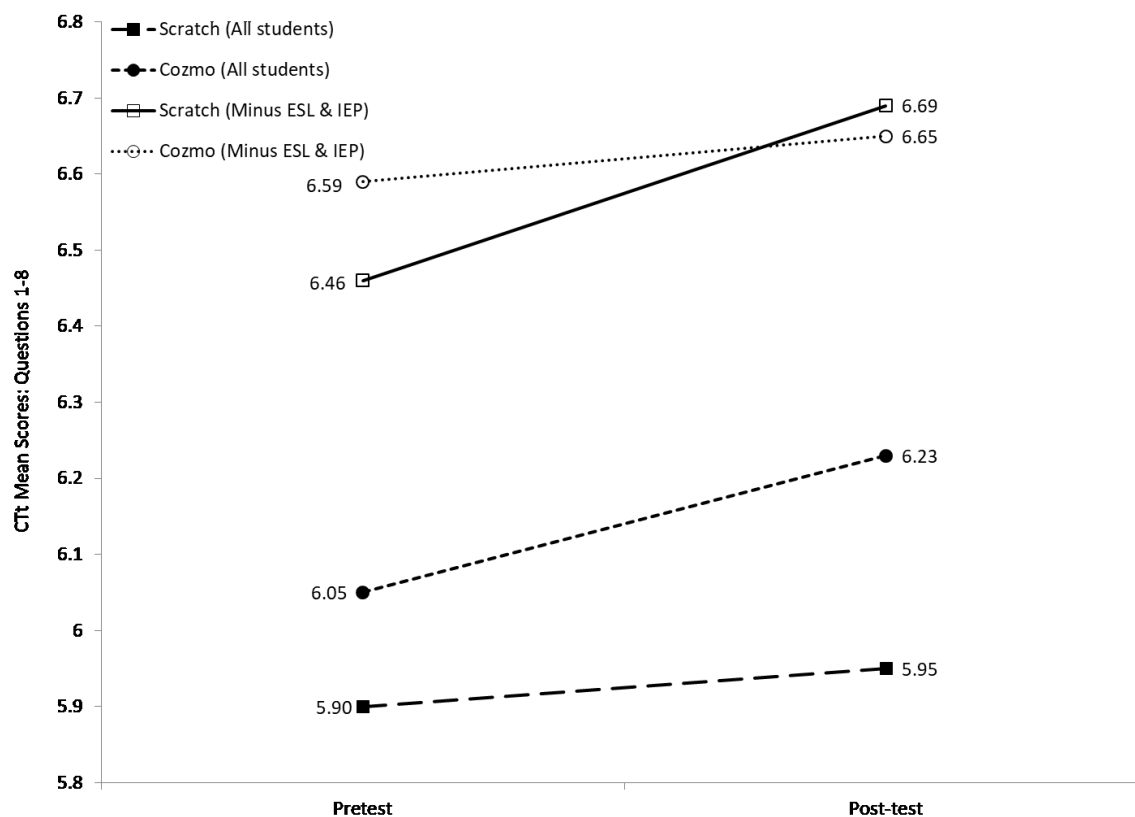
CTt: Questions 1-28 (Minus ESL & IEP). There was a non-significant main effect of intervention, *F* (1, 28) = 0.839, *p* = .367, indicating insufficient evidence that computational thinking skills were different between the Scratch and Cozmo groups. There was a significant main effect of time, *F* (1, 28) = 4.692, *p* = .039, indicating that computational thinking skills improved significantly from pre- to post-test for both the

Scratch and Cozmo groups. There was a non-significant interaction between time and intervention, $F(1, 28) = 0.012$, $p = .915$. This effect indicates that students in both interventions improved their computational thinking skills similarly over time. Specifically, scores were similar between the Scratch (pretest, $M = 18.46$, $SD = 3.53$; post-test, $M = 19.85$, $SD = 4.47$) and Cozmo (pretest, $M = 17.00$, $SD = 5.45$; post-test, $M = 18.53$, $SD = 4.11$) groups (see Figure 18). There seems to be a small effect size between the pre- and post-test computational thinking measures for Scratch ($d = .35$) and Cozmo ($d = .32$), as well as between the two learning conditions relative to the post-test score ($d = -.31$).
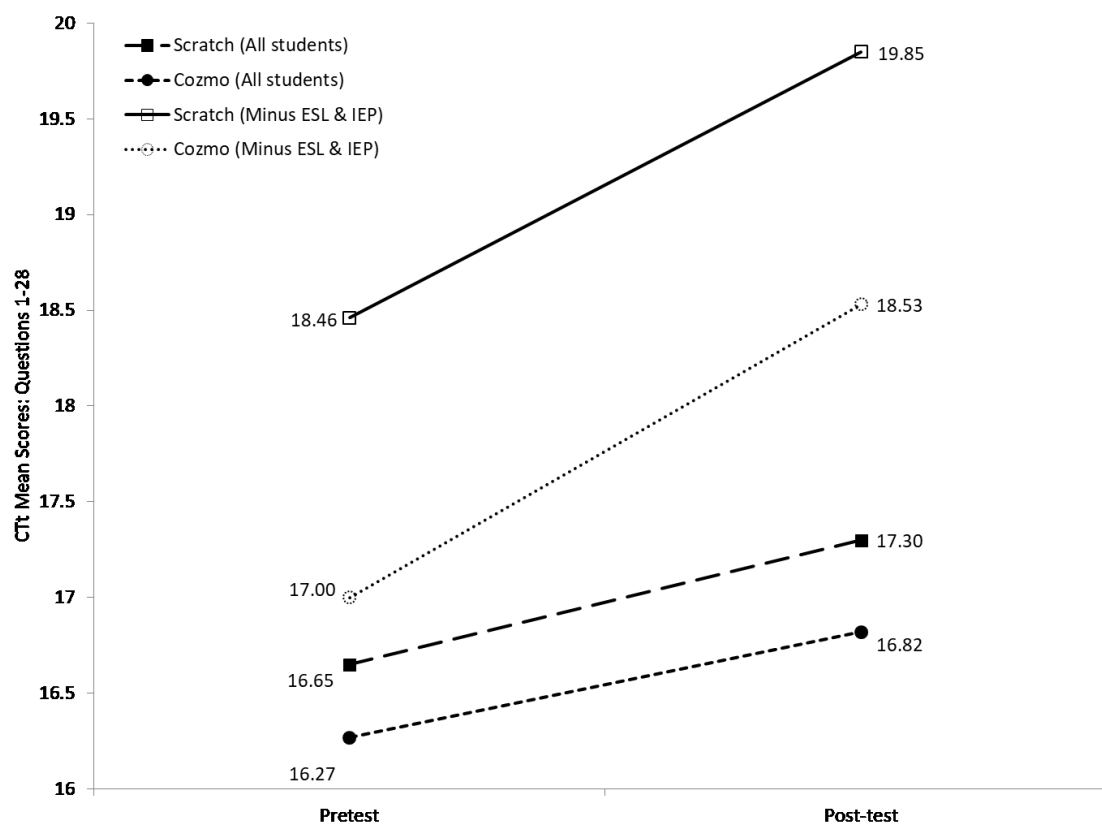
Figure 18. CTt mean scores: Questions 1-28.

CTt: Questions 1-8 (All students). Student scores for questions 1-8 were calculated separately because the experimental curriculum adapted for this study directly covers the computational concepts of basic directions, sequencing, and loops (repeat times) which aligns it with questions 1-8 on the CTt. There was a non-significant main effect of intervention, $F (1, 40) = 0.197$, $p = .660$, indicating insufficient evidence that computational thinking skills were different between the Scratch and Cozmo groups. There was a non-significant main effect of time, $F (1, 40) = 0.533$, $p = .469$, indicating that performance on the Computational Thinking Test was generally the same for the Scratch and Cozmo groups from pretest to post-test. There was a non-significant interaction between time and intervention, $F (1, 40) = 0.172$, $p = .680$. This effect indicates that both interventions had the same effect on computational thinking skills. Specifically, scores were similar between the Scratch (pretest, $M = 5.90$, $SD = 1.37$; post-test, $M = 5.95$, $SD = 1.28$) and Cozmo (pretest, $M = 6.05$, $SD = 2.01$; post-test, $M = 6.23$, $SD = 1.69$) groups (see Figure 19). There seems to be a very small effect size between the pre- and post-test computational thinking measures for Scratch ($d = .04$) and Cozmo ($d = .10$), as well as between the two learning conditions relative to the post-test score ($d = .19$).
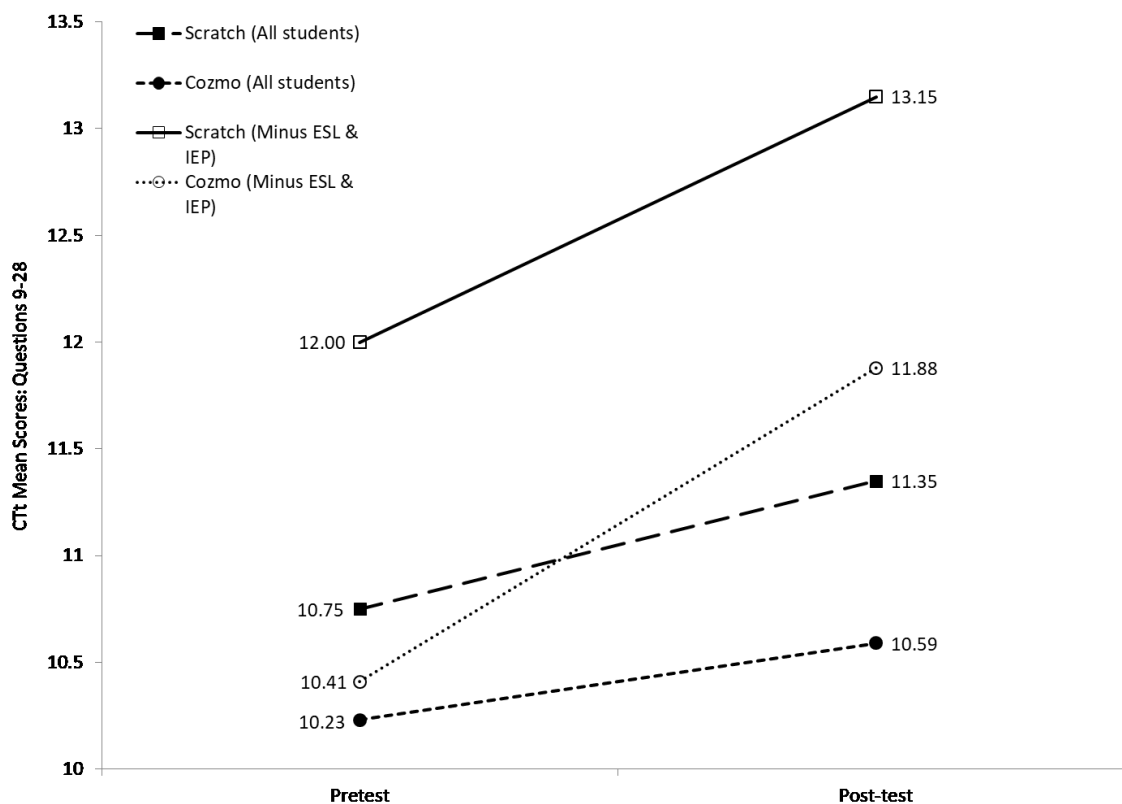
CTt: Questions 1-8 (Minus ESL & IEP). There was a non-significant main effect of intervention, $F(1, 28) = 0.010$, $p = .923$, indicating insufficient evidence that computational thinking skills were different between the Scratch and Cozmo groups. There was a non-significant main effect of time, $F(1, 28) = .591$, $p = .448$, indicating that performance on the Computational Thinking Test was generally the same for the Scratch and Cozmo groups from pretest to post-test. There was a non-significant interaction between time and intervention, $F(1, 28) = 0.208$, $p = .651$. This effect indicates that both interventions had the same effect on computational thinking skills. Specifically, scores were similar between the Scratch (pretest, $M = 6.46$, $SD = .88$; post-test, $M = 6.69$, $SD = .75$) and Cozmo (pretest, $M = 6.59$, $SD = 1.62$; post-test, $M = 6.65$, $SD = 1.32$) groups (see Figure 19). There seems to be a very small effect size between the pre- and post-test computational thinking measures for Scratch ($d = .28$) and Cozmo ($d = .04$), as well as between the two learning conditions relative to the post-test score ($d = -.04$).

Figure 19. CTt mean scores: Questions 1-8.

CTt: Questions 9-28 (All students). There was a non-significant main effect of intervention, F (1, 40) = 0.372, p = .546, indicating insufficient evidence that computational thinking skills were different between the Scratch and Cozmo groups. There was a non-significant main effect of time, F (1, 40) = .814, p = .372, indicating that performance on the Computational Thinking Test was generally the same for the Scratch and Cozmo groups from pretest to post-test. There was a non-significant interaction between time and intervention, F (1, 40) = 0.049, p = .826. This effect indicates that both interventions had the same effect on computational thinking skills. Specifically, scores were similar between the Scratch (pretest, M = 10.75, SD = 3.21; post-test, M = 11.35,

SD = 4.25) and Cozmo (pretest, M = 10.23, SD = 3.64; post-test, M = 10.59, SD = 4.08) groups (see Figure 20). There seems to be a very small effect size between the pre- and post-test computational thinking measures for Scratch (d =.16) and Cozmo (d =.09), as well as between the two learning conditions relative to the post-test score (d = -.18).

CTt: Questions 9-28 (Minus ESL & IEP). There was a non-significant main effect of intervention, $F (1, 28) = 1.402$, $p = .246$, indicating insufficient evidence that computational thinking skills were different between the Scratch and Cozmo groups. There was a significant main effect of time, $F (1, 28) = 4.867$, $p = .036$, indicating that computational thinking skills improved significantly from pre- to post-test for both the Scratch and Cozmo groups. There was a non-significant interaction between time and intervention, $F (1, 28) = 0.071$, $p = .792$. This effect indicates that students in both interventions improved their computational thinking skills similarly over time. Specifically, scores were similar between the Scratch (pretest, $M = 12.00$, $SD = 3.00$; post-test, $M = 13.15$, $SD = 4.00$) and Cozmo (pretest, $M = 10.41$, $SD = 4.12$; post-test, $M = 11.88$, $SD = 3.31$) groups (see Figure 20). There seems to be a small effect size between the pre- and post-test computational thinking measures for Scratch ($d =.33$) and Cozmo ($d = .39$), as well as between the two learning conditions relative to the post-test score ($d =-.35$).

Figure 20. CTt mean scores: Questions 9-28.

Spatial Skills

A repeated measures analysis of variance with a between-subjects factor (Cozmo versus Scratch) and within-subjects factor (MRT pre- and post-test) was performed to test the intervention's effects on spatial skills.

MRT: All students. There was a non-significant main effect of intervention, F $(1, 40) = 0.258$, $p = .615$, indicating insufficient evidence that spatial skills were different between the Scratch and Cozmo groups. There was a significant main effect of time, F $(1, 40) = 54.202$, $p = .000$, indicating that spatial skills improved significantly from pre-

to post-test for both the Scratch and Cozmo groups. There was a non-significant interaction between time and intervention, F (1, 40) = 1.751, p = .193. This effect indicates that students in both interventions improved their spatial skills similarly over time. Specifically, scores were similar between the Scratch (pretest, M = 7.55, SD = 4.67; post-test, M = 10.90, SD = 6.25) and Cozmo (pretest, M = 6.09, SD = 4.10; post-test, M = 10.91, SD = 4.67) groups (see Figure 21). There seems to be a very small effect size between the two learning conditions relative to the post-test score (d = .002). However, there was a medium effect size between the pre- and post-test spatial skills measures for Scratch (d = .61) and a large effect size for Cozmo (d = 1.10).

MRT: Minus ESL and IEP. There was a non-significant main effect of intervention, F (1, 28) = 1.437, p = .241, indicating insufficient evidence that spatial skills were different between the Scratch and Cozmo groups. There was a significant main effect of time, F (1, 28) = 34.675, p = .000, indicating that spatial skills improved significantly from pre- to post-test for both the Scratch and Cozmo groups. There was a non-significant interaction between time and intervention, F (1, 28) = 1.519, p = .228. This effect indicates that students in both interventions improved their spatial skills similarly over time. Specifically, scores were similar between the Scratch (pretest, M = 9.54, SD = 3.93; post-test, M = 12.77, SD = 6.07) and Cozmo (pretest, M = 6.71, SD = 4.36; post-test, M = 11.65, SD = 4.94) groups (see Figure 21). There seems to be a small effect size between the two learning conditions relative to the post-test score (d = -.20). However, there was a medium effect size between the pre- and post-test spatial skills measures for Scratch (d = .63) and a large effect size for Cozmo (d = 1.06).
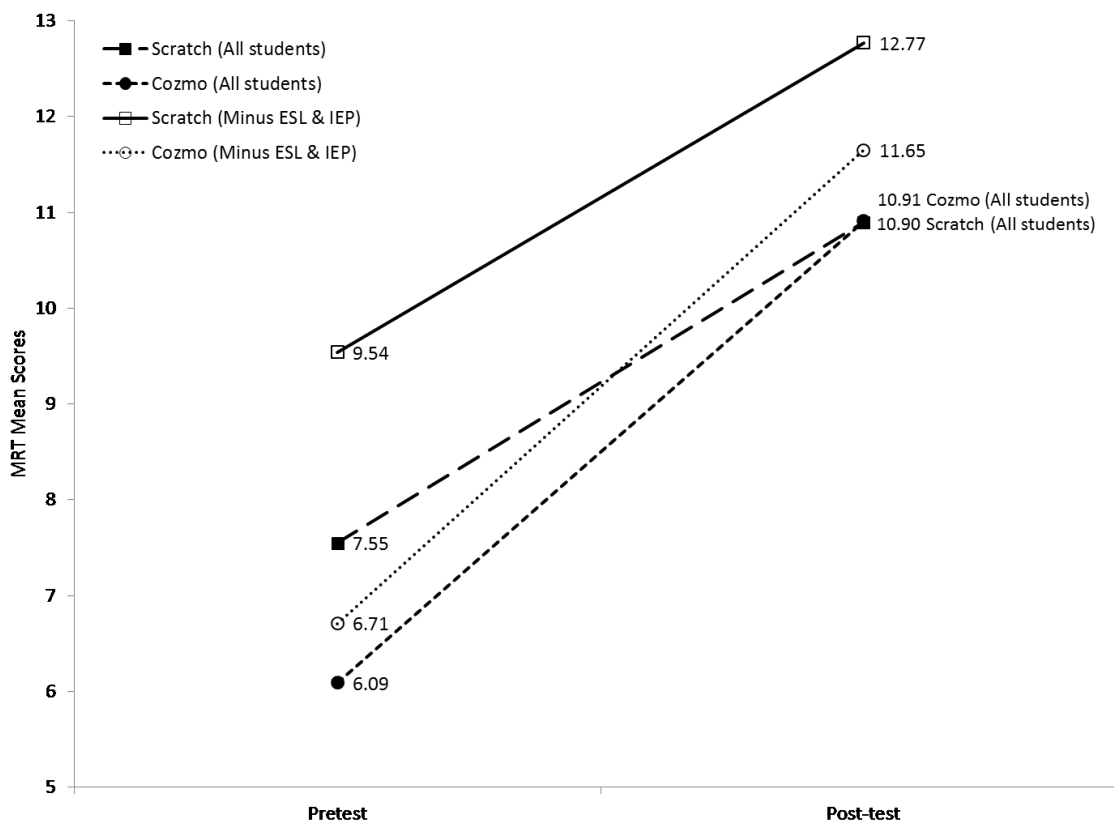
Figure 21. MRT means scores.

Competency Beliefs

A repeated measures analysis of variance with a between-subjects factor (Cozmo versus Scratch) and within-subjects factor (Competency Beliefs pre- and post-test) was performed to test the intervention's effects on competency beliefs.

Competency beliefs: All students. There was a non-significant main effect of intervention, $F(1, 40) = 0.012$, $p = .914$, indicating insufficient evidence that competency beliefs were different between the Scratch and Cozmo groups. There was a significant main effect of time, $F(1, 40) = 10.058$, $p = .003$, indicating that competency beliefs

improved significantly from pre- to post-test for both the Scratch and Cozmo groups. There was a non-significant interaction between time and intervention, *F* (1, 40) = 0.001, *p* = .970. This effect indicates that students in both interventions improved their competency beliefs similarly over time. Specifically, scores were similar between the Scratch (pretest, *M* = 30.70, *SD* = 6.66; post-test, *M* = 33.85, *SD* = 7.61) and Cozmo (pretest, *M* = 30.86, *SD* = 6.62; post-test, *M* = 34.09, *SD* = 6.57) groups (see Figure 22). There seems to be a small-medium effect size between the pre- and post-test competency beliefs measures for Scratch (*d* = .44) and Cozmo (*d* = .49), and a very small effect between the two learning conditions relative to the post-test score (*d* = .03).

Competency Beliefs: Minus ESL and IEP. There was a non-significant main effect of intervention, F (1, 28) = 0.202, p = .656, indicating insufficient evidence that competency beliefs were different between the Scratch and Cozmo groups. There was a significant main effect of time, F (1, 28) = 13.563, p = .001, indicating that competency beliefs improved significantly from pre- to post-test for both the Scratch and Cozmo groups. There was a non-significant interaction between time and intervention, F (1, 28) = 0.058, p = .811. This effect indicates that students in both interventions improved their competency beliefs similarly over time. Specifically, scores were similar between the Scratch (pretest, M = 32.00, SD = 6.61; post-test, M = 36.23, SD =7.40) and Cozmo (pretest, M = 30.71, SD = 6.90; post-test, M = 35.53, SD = 6.66) groups (see Figure 22). There seems to be a medium effect size between the pre- and post-test competency beliefs measures for Scratch (d = .60) and Cozmo (d = .71), and a very small effect between the two learning conditions relative to the post-test score (d = -.10).
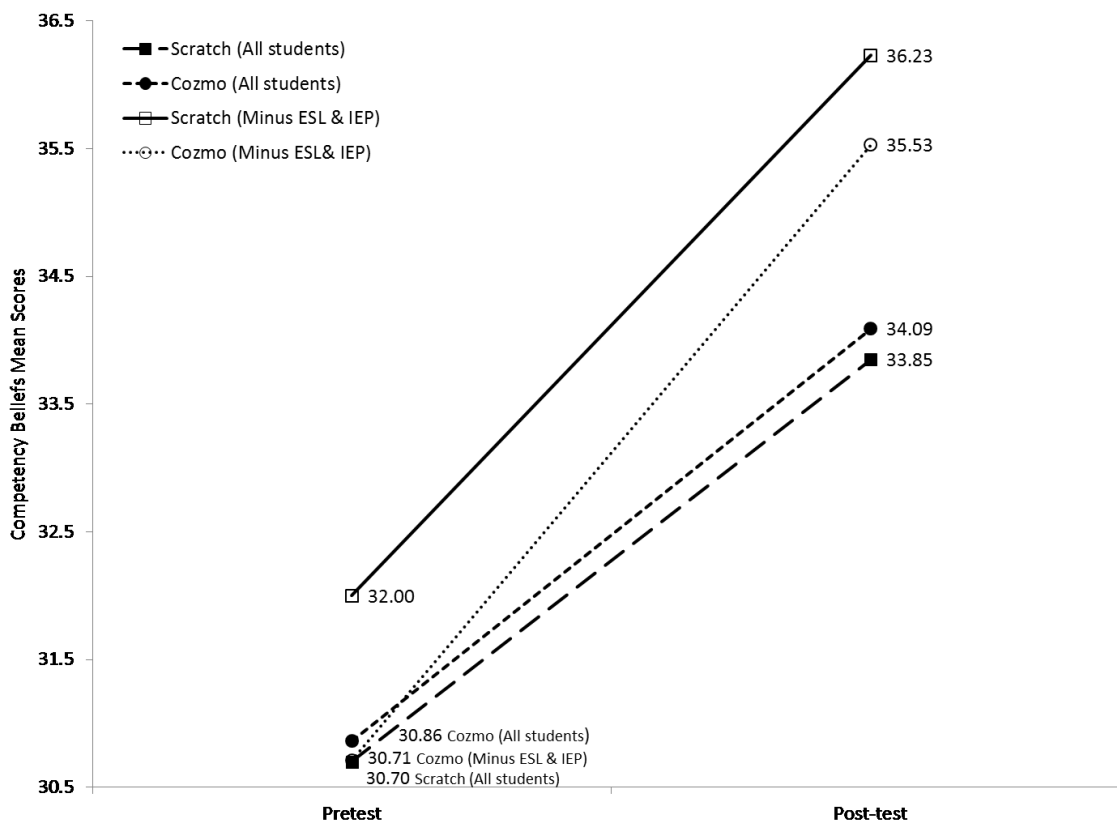
Figure 22. Competency beliefs mean scores.

Engagement

Engagement was measured each day of the intervention. The use of repeated measures ANOVA would capture data for only 15 students in the Scratch group (71% of the class) and 13 students in the Cozmo group (59% of the class) due to student absences throughout the intervention period. Multilevel models do not require complete data sets so whole cases do not need to be deleted when data are missing from one time point; instead, multilevel models make it possible for parameters to be estimated successfully with the available data (Field, 2013). In order to avoid the loss of data and provide a

clearer picture of daily engagement, a linear mixed model was performed to test the intervention's effects on engagement. Intervention (Scratch versus Cozmo), class period, and the interaction of intervention by class period were set as the fixed factors, and class period was set as the repeated factor with compound symmetry covariance structure.

Engagement: All Students. Treatment significantly predicted engagement, F (1, 41.492) = 4.654, p = .037, class period significantly predicted engagement F (8, 290.547), = 4.965, p = .000, and the interaction of treatment and class period, F (7, 290.481) = 2.380, p = .022, significantly predicted engagement. After controlling for the class period and interaction, these analyses showed that intervention significantly predicted engagement, b = -.320, t (114.228) = -2.017, p = .046. The Cozmo group was significantly more engaged than the Scratch group with a medium effect (d = .45). Pairwise comparisons showed that students found class period 3 significantly more engaging than period 2 (p = .000), period 5 (p = .029), period 6 (p = .000), period 7 (p = .000), period 8 (p = .007), and period 9 (p = .001). The Cozmo group also found period 3 significantly more engaging than the make-up day (p = .022). Students also found class period 4 significantly more engaging than period 2 (p = .028), period 6 (p = .027), and period 7 (p = .001). Additionally, students found class period 5 significantly more engaging than period 7 (p = .003). Finally, students found class period 8 significantly more engaging than class period 7 (p = .015). Pairwise comparisons for the interaction of class period by intervention showed that engagement varied between the Scratch and Cozmo groups during class period 3 (p = .002), period 4 (p = .015), and period 9 (p =

.046). See Figure 23 for a summary of mean engagement scores during each day of the intervention.

Engagement: Minus ESL and IEP. Treatment significantly predicted engagement, $F(1, 29.544) = 6.963$, $p = .013$, class period significantly predicted engagement $F(8, 205.462), = 3.550$, $p = .001$, and the interaction of treatment and class period, $F(7, 205.516) = 1.152$, $p = .332$, did not significantly predict engagement. After controlling for the class period and interaction, these analyses showed that intervention significantly predicted engagement, $b = -.433$, $t(97.243) = -2.318$, $p = .023$. The Cozmo group was significantly more engaged than the Scratch group with a medium effect ($d = .63$). Pairwise comparisons showed that students found class period 3 significantly more engaging than period 2 ($p = .001$), period 6 ($p = .001$), period 7 ($p = .002$), period 8 ($p = .034$), and period 9 ($p = .014$). The Cozmo group also found class period 3 significantly more engaging than the make-up period ($p = .019$). Students found class period 4 significantly more engaging than period 2 ($p = .041$) and period 6 ($p = .046$). Additionally, students found class period 5 significantly more engaging than period 2 ($p = .048$). See Figure 23 for a summary of mean engagement scores during each day of the intervention.
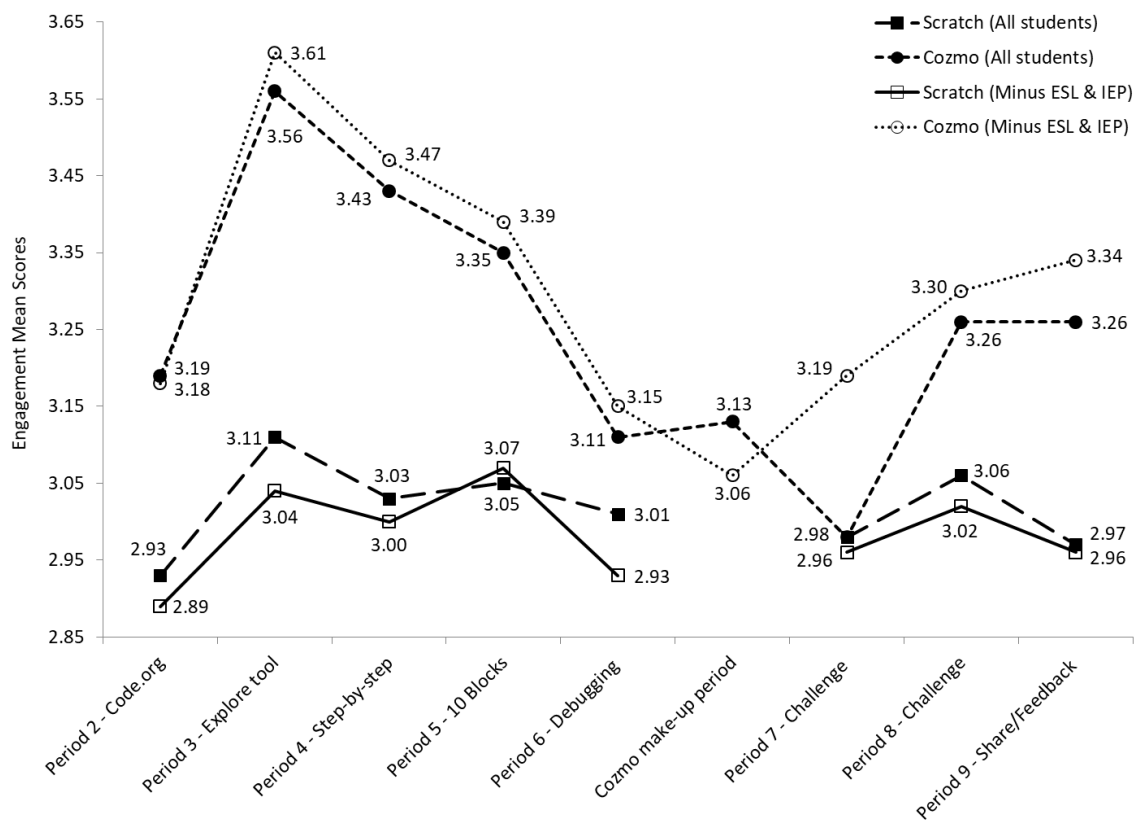
Figure 23. Engagement mean scores.

CHAPTER V

DISCUSSION, IMPLICATIONS, AND RECOMMENDATIONS

Discussion

The purpose of this study was to examine the effects of the coding activities supported by the artificially intelligent, animated emotional-educational robot Cozmo on middle school students' computational thinking, spatial skills, competency beliefs and engagement compared to the more traditional computer-based program of Scratch. Two separate analyses were conducted for each dependent variable: one for all students and one that excluded scores for students with special education needs (IEP) and with English as a Second Language (ESL) because comprehension of instructions and questions on each assessment was an issue for these students. The study took place in a public school with a diverse group of students who did not self-select into the Technology class. This resulted in a mix of students with different background, interest, and ability levels.

Hypothesis 1

The hypothesis that Cozmo would produce greater gains in computational thinking was not supported by the data from the Computational Thinking Test (CTt). These results are similar to other studies that found no significant difference in computational thinking skills when comparing interventions using computer-based and physical robots (Berland & Wilensky, 2015; Djambong & Freiman, 2016; Kazakoff & Bers, 2012).

163

There was not a significant difference in pre- and post-test scores for questions 1-8 that were directly aligned to the instructional unit that was adapted for the study. Descriptively, there was improvement from pre- to post-test for both groups on questions 1-8, but the small number of questions may have contributed to the difference not reaching significance. There was significant improvement on questions 9-28 for both the Scratch and Cozmo groups from the beginning to the end of the study when removing scores for students with special education needs (IEP) and English as a Second Language (ESL) who had difficulty reading and comprehending the instructions and questions on the assessment. It is possible that students developed additional skills not directly covered in the instructional unit by completing the challenge projects at the end of the unit. Leonard et al. (2016) conducted a study in which two different methods were used for a gaming intervention: tutorial first then project completion versus project first where students came up with an idea for a game and then learned to code without a tutorial. The project first method had better results than the tutorial first method. This suggests that students learn skills in the context of coding for a project.

There was not a significant difference in pre- and post-test scores on all 28 questions of the Computational Thinking Test when all students were included in the analysis; however, there was statistically significant improvement for both the Scratch and Cozmo groups from the beginning to the end of the study when removing scores for students with special education needs (IEP) and English as a Second Language (ESL) who had difficulty reading and comprehending the instructions and questions on the assessment. Both intervention groups improved similarly from the beginning to the end

of the intervention, but there were multiple factors that may have influenced performance on the Computational Thinking Test in each of the classes.

Attendance was one factor that could have negatively influenced performance as absences remove students from instruction and practice that can improve their comprehension and skills. The Scratch group had seven students who were absent at least one day of the intervention period while the Cozmo group had ten students who were absent at least one day of the intervention period.

Cooperation was another factor that may have impacted performance on the CTt. Individual student accountability was higher in the Scratch group. These students were permitted to sit next to whomever they wished and encouraged to help one another as needed, but ultimately each student was responsible for completing the unit and final project independently. In the Cozmo group, students selected their own groups. Some students discovered as the intervention progressed that they were not able to work well with other group members and requested to switch groups; a common issue was one student dominating time with the iPad thus preventing the other students from fully participating in the lesson. This also resulted in some students floating from group to group throughout the intervention and reduced individual accountability for contributing to the activities and assignments.

Technical issues were an additional factor that could have affected the CTt scores. The Scratch group worked on the desktop computers in the lab where they have technology class daily and they experienced no technical difficulties throughout the

intervention. This was not the case for the Cozmo group that had to travel to the high school library each day in order to use the iPads housed there. On multiple days of the intervention, high school classes came in to the library to use the Chromebox lab causing a loss in bandwidth that resulted in the Cozmo app crashing for many of the groups. This negatively impacted performance because some students were not able to complete all of the lesson objectives due to the app not working. Even with one make-up day added for the Cozmo group, many students were still unable to complete all of the activities and assignments due to the accumulation of multiple days without full access to the app due to the technical difficulties.

The importance of time and an adequate amount of training time for skills development has been cited repeatedly in the literature (Atmatzidou & Demetriadis, 2016; Bers, Flannery, Kazakoff, & Sullivan, 2014; Clark, Tanner-Smith, & Killingsworth, 2015) and time was a concern in the current study in multiple ways. The instructional interventions implemented with both groups served as a brief introduction to programming (less than ten class periods) and the number of days originally planned for several of the activities had to be reduced due to circumstances beyond the researcher's control. It is also important to note that circumstances on the day the CTt and MRT post-tests were given differed for the Scratch group compared to the Cozmo group. The Scratch group had a full class period and was able to complete the Mental Rotations Test and the Computational Thinking Test without issue. The Cozmo group should also have had a full class period to complete the Mental Rotations Test and the Computational Thinking Test, but adverse weather conditions on the originally-scheduled day (which

was one day later than the Scratch group due to the added catch-up day caused by technical difficulties for the Cozmo group) resulted in a district-wide snow day. This moved Lesson 9 for the Cozmo group to the last day of the semester. The regular classroom teacher had to give a state-required assessment on that day because she would not see her students again (Technology is a semester-long class). This resulted in the following structure of the class period: the timed Mental Rotations Test was given first since it is a timed test and it must be given to the whole group simultaneously, the Computational Thinking Test was given second, and students were told to complete their technology assessment for the regular teacher when they finished the Computational Thinking Test. This resulted in many students rushing through the Computational Thinking Test (for which they were not receiving a grade) in order to complete the required technology assessment (for which they could receive an exam grade that would affect their overall grade for the class).

In spite of the obstacles, student performance on the CTt in the Cozmo group was comparable to the Scratch group. In fact, more students in the Cozmo group than the Scratch group improved their computational thinking scores from pre- to post-test (12 versus 10), and the highest gains could also be found in the Cozmo group (e.g., the largest gain was achieved by a female student who increased 10 points from pre- to post-test on the CTt).

Concern over how to assess student progress in programming has been expressed in a number of studies (Brennan & Resnick, 2012; G. Chen et al., 2017; Djambong & Freiman, 2016; Shuchi Grover et al., 2014; Román-González et al., 2017). The current

study conducted multiple methods of measurement to develop a clearer picture of student understanding of computational thinking. The debugging challenges that were included in the curriculum addressed student understanding as students had to identify and correct errors in code in order for it to run correctly. Additionally, rather than the researcher taking student code and inferring understanding from it, the study implemented a final project that utilized a rubric with programming concepts that had to be included and successfully executed in the given program (see Appendix B). Students shared these projects and received peer feedback that they used to make improvements to their projects (see Appendix C for examples of student projects).

The Computational Thinking Test that was used to assess computational thinking skills was the most labor intensive of all of the assessments conducted during the study. The authors that created and validated the instrument explained that when reliability was studied with regard to grade and administration, reliability increased as grade level increased, and reliability increased when the CTt was administered through mobile devices; perhaps because they allowed the user to rotate the screen thus reducing the spatial cognitive load of the items (Román-González, Pérez-González, & Jiménez-Fernández, 2017, p. 9). Students in the current study were administered a paper version of the CTt and may not have realized that they could rotate the paper as needed in order to reduce the spatial cognitive load of the items. Djambong and Freiman (2016) also pointed out that pencil/paper assessments can be demotivating for students who are used to using computers in a course and students can randomly guess on multiple choice questions and it is not possible to tell their thought process. The modality of the

instrument has also been found to impact performance and boys have been found to be more reluctant writers than girls (Atmatzidou & Demetriadis, 2016; Merisuo-Storm, 2006).

Denning (2017) argued that computational thinking is a skill to be practiced and that performance-based assessments can give a clearer picture of competence. This aligns with Papert's (1980) discussion of the drudgery of school and standardized tests. Assessments containing questions with one prescribed correct answer do not encourage students to think through problems, to act them out, to authentically apply their skills in a given situation. In the rushed state the Cozmo group took their CT post-test, it is possible and probable that many students did not even read the questions and just bubbled in their answers (as evidenced by students flipping through the pages and submitting the test within a matter of minutes). This does not provide an accurate picture of their true performance potential. It is also possible that Scratch is better aligned with the CTt because they both use two-dimensional images.

It has been suggested that computational thinking is possibly related to general academic ability based on findings that classes with more high-performing students outperformed all other classes and low-performing students scored the lowest (G. Chen et al., 2017). Comprehension certainly plays an important role in performance as evidenced in the current study where removing students with special education needs and English as a Second Language resulted in significant improvement from pre- to post-test while including them in the analysis did not result in significant improvement. It is impossible

to determine the degree to which lack of time, comprehension, effort, and engagement influenced the results.

Hypothesis 2

The hypothesis that Cozmo would produce greater learning gains in spatial skills was not supported because the spatial skills of both groups improved significantly from pre- to post-test; however, the effect size was larger for the Cozmo group. Descriptively, the Cozmo group outperformed the Scratch group. More students in the Cozmo group than the Scratch group improved their spatial skills scores from pre- to post-test (20 versus 14), and the highest gains could also be found in the Cozmo group (e.g., the largest gain was achieved by a female student whose score increased 11 points from pre- to post-test on the MRT).

These results are promising considering studies in the literature that report significant improvement in spatial abilities with robotics interventions compared to groups that did not participate in comparative activities. For example, Julià and Antolí (2016) found a statistically significant greater increase in spatial ability post-test mean scores for students participating in a robotics course, but the comparison group did not receive similar activities and the difference between pre- and post-test means was not statistically significant for either group. Similarly, Coxon (2012) implemented a simulated robotics competition with one group of gifted children compared with a control group of gifted children that did nothing comparable and found that experimental group males evidenced significant and meaningful gains in measured spatial ability (Cohen's *d*

= .87) but experimental group females did not evidence significant gains. Two of the three subtests (two-dimensional and three-dimensional) in the assessment demanded mental rotations and females did not demonstrate significant improvement in either of those measures.

The current study used a test of mental rotation skills and both the Scratch and Cozmo groups showed significant improvement from pretest to post-test as well as practically meaningful improvement with the Cozmo group having a larger effect. Especially encouraging is the fact both of these classes included a heterogeneous sample of public school students with a higher female to male ratio. The Code Lab app used to program Cozmo uses an open-source version of the visual programming language Scratch called Scratch Blocks which makes coding with Cozmo and Scratch similar. Using the same curriculum with the same programming language yielded significant improvement with both the Scratch and Cozmo groups, but the larger effect with Cozmo suggests the added benefit of a physical robot making the program concrete in three-dimensional reality compared to the two-dimensional rendering on a flat screen with Scratch. This aligns with the findings of Fesakis et al. (2013) and Kalelioglu (2015) that students using virtual environments can have difficulty giving directional commands when programming objects (e.g., looking for a 'down' arrow when trying to move a sprite on the screen). Low spatial skills can present challenges to novices learning to program and can lead to the Catch-22 described by Uttal & Cohen (2012) of students not having the knowledge that would allow them to succeed in STEM fields despite relatively low spatial skills because they cannot get through the early STEM classes

where they must rely on their spatial abilities. The finding of improved mental rotation skills in the current study is reassuring because spatial abilities can predict STEM attainment and achievement (Wai, Lubinski, & Benbow, 2009).

Hypothesis 3

The third hypothesis was that on average, both the Cozmo and Scratch training groups were expected to experience an increase in competency beliefs. This hypothesis was confirmed because the competency beliefs of both groups increased significantly from the beginning to the end of the intervention. The effect size was medium for both groups, but the effect was larger for the Cozmo group. The highest gains could also be found in the Cozmo group. Double-digit gains were made in both groups, but there were more of these scores in the Cozmo group: In the Scratch group, two students increased 10 points while in the Cozmo group one student increased 10 points, one student increased 13 points, one student increased 14 points, and one student increased 25 points.

Beliefs about ability in STEM disciplines can be more predictive of performance than prior experience and outcome expectations, and gifted women may be more prone to underconfidence in the traditionally male-dominated STEM fields (Pajares, 1996; Zeldin & Pajares, 2000). Thus, the improved competency beliefs found in both the Scratch and Cozmo groups in this study are promising. These findings are also contrary to the findings by Weese et al. (2016) that student self-efficacy was higher initially (i.e., pre-intervention) due to overconfidence. Leonard et al. (2016) also found that pre- to post-self-efficacy scores for fifth and eighth graders on the construct of computer use declined

significantly after using LEGO EV3 robots and MINDSTORMS to program them. Witherspoon, Schunn, Higashi, and Shoop (2018) using pre and post scores in interest, identity, and competency beliefs showed pre-post declines but to different degrees; and Lewis (2010) found that students who used the Logo programming language felt more confident about their competence writing programs than students using Scratch. A difference in the instruments used to assess self-efficacy may have influenced the different outcomes in these studies. Both groups in the current study significantly increased their competency beliefs. While this was the initial exposure of both groups to programming, the use of the same visual programming language may have contributed to these findings because it reduced the cognitive load compared to that required when using a more complex text-based programming language, thus allowing students to achieve "small wins" (Witherspoon et al., 2018) and positively affecting their competency beliefs.

These findings are similar to those found in the literature showing that both robotics (Hinton, 2018) and Scratch (Joshua Levi Weese, 2016) improved student self-efficacy. Y.-F. Chen, Cannady, Schunn, and Dorph (2017), the authors of the instrument adapted to measure competency beliefs in this study, conceived competency beliefs as being semi-malleable and amenable to intervention, but cautioned that changes in scale scores are not to be expected due to single hour long experiences (i.e., it is more effective with longer interventions). Nugent et al. (2015) found that a longer 40-hour robotics intervention (summer camp building and programming robots) resulted in greater self-efficacy to perform robotics tasks and significantly greater confidence in abilities

compared to students in a shorter three-hour intervention (20 minute rotations at learning stations teaching different concepts), although these students did have significant increases from pretest to post-test in their self-efficacy with robotics. The current study's intermediate time period resulted in significant increases in self-efficacy for both groups with practically meaningful effect sizes and a larger effect for the Cozmo group. Nugent et al. conducted their interventions at a STEM summer camp with students already interested in the field whereas the current study took place in a public middle school classroom, thus findings of increased self-efficacy and engagement found in the current study are especially reassuring.

Hypothesis 4

The fourth hypothesis was that Cozmo was expected to produce greater student engagement. This hypothesis was confirmed by the finding that the Cozmo group was significantly more engaged than the Scratch group with a medium effect. There were several factors that may have influenced student engagement throughout the study.

Attendance was a larger concern for the Cozmo group. The Scratch group had seven students who were absent at least one day of the intervention period while the Cozmo group had ten students who were absent at least one day of the intervention period. This did not have as large of an effect on the Scratch students because they were working independently in their own accounts and could make up any missed activities before proceeding. The impact was greater when students were absent in the Cozmo group because students were working in groups and when absent students returned to

class they found themselves behind because the rest of the members of the group had already completed the previous day's activities and could not go back and redo them to catch the person up.

Personal issues are another factor that can affect student engagement and seemed to be a larger concern in the Cozmo group. For example, during period six of the intervention, two students came to the researcher in tears and asked to see the guidance counselor about something that had happened outside of class. These two students missed the rest of the class period and were both absent the following day. When they returned to school during period eight, they were still upset about the personal issue and their engagement scores were lower than they had been before the personal issue began. During period nine, they seemed to be managing the personal issue better (e.g., there were no tears and/or asking to use the restroom or see the guidance counselor) and their engagement scores increased.

Technical issues may also affect students' impressions (Sarmento, Reis, Zaramella, Almeida, & Tacla, 2015) and that appeared to be the case in the current study. The Scratch group experienced no technical issues and their engagement scores were similar throughout the intervention. However, a trend appears when observing the pattern of daily engagement for the Cozmo group: engagement steadily declines on each of the consecutive days when the app was crashing due to multiple classes decreasing the available bandwidth. One would expect engagement scores to continue to decline if this was an indication of loss of overall engagement, but engagement scores increased after the technology began working again.

While many studies have shown that the use of physical robots has a positive effect on student engagement (Atmatzidou & Demetriadis, 2016; Chang et al., 2010; Chin et al., 2014; Highfield, 2010; Karim et al., 2016; Klassner & Anderson, 2003; Lauwers, Nourbakhsh, & Hamner, 2009; Merino-Armero et al., 2018; Mitnik, Nussbaum, et al., 2009; Nourbakhsh et al., 2004; Nugent et al., 2010; Phetsrikran et al., 2017; Poh et al., 2016; Robinson, 2005; Rogers & Portsmore, 2004; Sutton & Williams, 2007b; Wei et al., 2011; Witherspoon et al., 2017), others have not. For example, Hussain, Lindh, and Shukur (2006) found no general positive attitude toward LEGO robots, and that pupils with higher math ability tended to be more engaged and had a positive attitude toward LEGO material compared to other students. More recently, Pugnali and Sullivan (2017) found no significant difference in engagement or collaboration scores between a tangible/robotic interface (KIBO) and a graphical interface (ScratchJr).

The written reflections collected from students at the end of the intervention in the current study supported the finding that Cozmo was more engaging than Scratch. Feedback from the Scratch group was mixed. One question in particular asked what students might want to do next. Some students in the Scratch group expressed interest in continuing with coding, but others did not. Responses including "Nothing involving coding" and "Take a break from coding" are the opposite desired reactions when the goal is for students to want to continue learning more.

Additional comments included "Scratch is not my favorite," "Scratch is boring," and "I would like to try something less boring than Scratch." This differed greatly from the Cozmo group where no negative feedback was received. The feedback about Cozmo

was positive and included students expressing the desire to see what else Cozmo could do and to make more complex projects. Several students commented that they wanted to buy their own Cozmo, one student wanted to keep Cozmo, and one student wrote "Cozmo is cute and I love him." The researcher observed disappointment from the Cozmo group that it was time for Cozmo to leave in audible groans when it was time to put Cozmo away, comments that they would like to continue, and one student holding Cozmo in front of her face and telling Cozmo she was sad he had to leave and she would miss him. Cozmo is an autonomous emotional-educational robot with additional features that encourage interaction including artificial intelligence, animation and emotions, facial recognition, and augmented reality capabilities made possible through computer vision. This high level of interactivity did result in a higher level of engagement for students using Cozmo compared to students working with the two-dimensional Scratch characters on a computer screen. This finding is supported by Wainer et al. (2006) who found that people rated a physical robot as more enjoyable than a simulated robot.

Limitations

There were several limitations that should be noted in this study. Conducting the study in a public school environment required the use of intact classes which threatens the internal validity of the study. Several efforts were made to attempt to establish the degree of equivalence between the two treatment groups including holding variables constant (e.g., same school, same grade level, same semester) and collecting and reporting the same demographical information for all participants. Additionally, the pretest-posttest, nonequivalent control group design used in the study aided in checking

the extent of group similarity, and the pretest scores were used for statistical control and for generating gain scores. However, the pretest-posttest design introduced testing threat since the initial administration of the Computational Thinking Test, Mental Rotations Test, and Competency Beliefs survey could cue students to desirable responses on the final administration of each instrument (i.e. pretest sensitization). However, the extended time between administrations of the tests helped mitigate this effect. The pretests were given before winter break in mid-December, and the intervention did not begin until after winter break in January with the post-test following in mid-January. Low statistical power may have been an additional limitation as evidenced by non-significant p values despite moderate effect sizes on several of the pretest data analyses (e.g., Tables 13, 19, 20) as well as spatial skills and competency beliefs.

The use of closed-ended questions on the Computational Thinking Test can be considered a limitation because students could randomly guess their answers, and it was not possible to observe student thought processes leading to selected responses. And as with any study that utilizes surveys as an instrument, there was the threat of recall bias for the Competency Beliefs survey and the Engagement survey that could affect the internal validity of the study. While the selected instruments have their drawbacks, each instrument was selected because of the acceptable reliability statistics reported in the literature and to make it possible for other researchers to replicate the study, thus strengthening its external reliability.

The timing of the intervention (the end of a semester long class) and technical issues were additional limitations. Technical issues that resulted from the Code Lab app

crashing as a result of reduced bandwidth from multiple classes using the Wi-Fi in the library and snow days resulted in students in the Cozmo group being unable to complete all of the activities and having the full amount of time needed to complete all of the post-test assessments.

Delimitations

The experimenter effect of testing was a threat to external validity because students may have answered the way they thought the researcher wanted them to answer. The results also may not be generalized beyond the sample of seventh grade students enrolled in technology class at the public middle school in the Midwestern United States. Additionally, while the Scratch group completed the intervention in the Technology classroom that offered a quiet atmosphere with enough computers for each student, a projector for instruction, and the capability to control student screens to get attention for instruction and demonstration; none of these affordances were available in the Cozmo setting. The Cozmo intervention took place in the high school library where there were continually students entering and exiting adding to the noise and distraction level, there was no projector available for instruction and no capability to take over student screens for instruction and demonstration, and an entire English class came in during the three main instructional days of the intervention to use the Chromebox lab causing a reduction in bandwidth resulting in the Cozmo app crashing repeatedly throughout all three class periods. Thus, results of this study may not generalize to a quieter computer laboratory or classroom setting.

Implications

This study helped address the noted lack of research on the integration of robotics in the classroom (Benitti & Barreto, 2012; Poh et al., 2016). Gaudiello and Zibetti (2016) found that learning by robotics has strong positive impacts on the affective, social, cognitive and metacognitive dimensions of learning and can profoundly transform student and teacher attitudes, but these effects are not the result of robotics-based activities alone and require the scaffolding provided by a suitable pedagogical approach. The current study adapted a curricular unit from the Creative Computing Course (Brennan et al., 2011) developed for Scratch to fit the blocks available in the Code Lab app and achieved positive results. The Scratch and Cozmo groups performed similarly and both groups significantly increased their spatial skills (all students and without ESL and IEP), competency beliefs (all students and without ESL and IEP), and computational thinking (without ESL and IEP only) from pretest to post-test. This shows that the scaffolded approach to programming instruction utilized in the Creative Computing Course was effective and can be adapted to fit other and new programming languages and technologies as they are developed. These findings can also support teachers in search of tools, curriculum, and pedagogy to inform classroom practice and can be adapted to different tools, academic disciplines, and needs. Curriculum designers can also benefit from incorporating scaffolded instructional techniques when designing new courses. The significant improvement attained in this study also lends support to initiatives and policy aimed at including and integrating technology in the curriculum.

Clarifying a pedagogical approach that is effective for programming instruction is an important first step, but it is also important to consider the tools that can assist educators in meeting instructional goals. Cozmo has proven to be an effective tool to cognitively and affectively engage students in learning. Cozmo resulted in statistically significant increases in computational thinking, spatial skills, and competency beliefs. Cozmo also had a larger effect on student spatial skills and competency beliefs and it is unclear how computational thinking scores may have been affected had the Cozmo group been given the same amount of time for the post-test as the Scratch group. Students were also more engaged with Cozmo than Scratch. Papadakis et al. (2016) found that Scratch was less popular than App Inventor for Android. They advocated that Scratch may be most appropriate for teaching young students, early learners, or in curriculum where the main aim is surface rather than deep knowledge about programming. They recommended the following sequence as a way to teach programming and computing fundamentals: ScratchJr followed by Scratch followed by App Inventor for Android and ending with a conventional text-based programming language. One issue with this sequence is the lack of efficiency in learning different formats and programs along each step of the continuum. The results of the current study demonstrated that Cozmo has a low floor (easy to begin) because students who had never had any formal programming instruction were able to significantly increase their scores in computational thinking, spatial skills, and competency beliefs, and students were significantly more engaged with Cozmo. The results also showed that Cozmo has wide walls (supports many different types of projects to appeal to people with different interests) by appealing to all of the

students in a public education classroom including females and minority students, and resulting in a wide variety of project types in the final Challenge open-ended project.

Cozmo offers a higher ceiling (able to create more complex projects over time) than Scratch, App Inventor, and other programs because it offers the ability to transition from a visual to a text-based programming language using the same tool. The Code Lab App offers Sandbox Mode with horizontal grammar similar to ScratchJr (developed for children ages five through seven) that can be used to introduce students as young as elementary school to coding. It also offers Constructor Mode with vertical grammar similar to Scratch (developed for eight through sixteen year olds but can be used by people of all ages) that can be used with students in elementary through high school. What sets Cozmo apart from other options is the additional availability of the SDK (Software Development Kit) that uses Python, a professional text-based programming language. Not only does Cozmo offer the capability to learn programming with a text-based programming language, but its unique additional features (animation, artificial intelligence, computer vision, emotional-educational robotics) offer other avenues of study in the technology realm as evidenced by Cozmo being used to teach the first artificial intelligence curriculum being offered to middle school students (Martines, 2018). Thus, Cozmo is a versatile platform that can develop student programming competence from novice to professional.

Recommendations for future research

In-school robotics interventions are more likely to include students with a wide variety of interests which can make it more challenging to keep all students equally engaged in programming. One of the known challenges in implementing robotics courses in K-12 environments, unlike in informal robotics programs, is that students have not self-selected into these learning environments. The meaningful effects in computational thinking, spatial skills, competency beliefs, and engagement found for the public middle school students using Cozmo in the current study illuminate a fertile field for future research.

While this study has identified the positive effects that both Scratch and Cozmo can have on student affect and skills, additional research can provide further insights and enhance practical applications. Future studies should more closely investigate the effects of treatment on computational thinking with a particular focus on the effects of time and methods of assessment. The time constraints and differences in post-test administration times suggest that future studies should be implemented earlier in a course (i.e., not at the end of a semester) to allow time to make adjustments for unanticipated technical and weather-related issues that may arise and provide students with ample time to complete all activities and assessments. It is also important to investigate other methods of assessing computational thinking. Denning (2017) pointed out that there is no consensus on what constitutes the skill of computational thinking and current assessment methods are unreliable indicators. Assessing skill by performance (e.g., code-a-thons and projects) is becoming more popular and supported by professionals in the field (Denning,

2017). Brennan and Resnick (2012) also highlighted the drawback of using a single assessment for computational thinking and they described different approaches for assessment that could be used in combination including project portfolio analysis; artifact-based interviews; and design scenarios. Students with special education needs and English as a Second Language did not significantly improve their computational thinking as measured by the CTt. Assessments that focus more on evaluating programming skill and rely less on reading ability using multiple choice questions could provide a clearer picture of the effects of treatment. Investigating a variety of methods of assessment for computational thinking that are more closely aligned with skills addressed in the curriculum will also be beneficial since only eight out of 28 questions on the CTt addressed skills that were covered in the curriculum unit that was used.

Armoni et al. (2015) used Scratch for their intervention and suggested that positive results similar to theirs should be observed when adapting any environment to teach middle school students. The current study supported this conclusion by finding similar positive results when adapting a curricular unit developed for Scratch to the Code Lab app for Cozmo. Therefore, additional research using the Creative Computing Course would be a fruitful area of exploration. A single unit of the full curriculum was implemented in the current study as a context for introducing students to programming. Brennan et al. (2011) suggested that the course can be used with everyone in grades K-12 and beyond to promote creativity and computational thinking. Adapting the course for use in different grade levels and subject areas as well as with different technologies could

provide valuable information that can be used to guide technology integration across curriculum areas.

Longitudinal research should also be conducted to determine whether the gains earned in the short-term duration of the study are sustainable in longer interventions. This can be accomplished in a variety of ways. A semester-length or full year course could be implemented using the additional units available in the Creative Computing Course. Additionally, studying the transition from horizontal grammar (Sandbox Mode) to vertical grammar (Constructor Mode), and then from a visual programming language (Scratch Blocks) to a text-based programming language (Python) using a single robotics platform was not possible until the emergence of Cozmo. Investigating the application of Cozmo as a transitional tool to increasingly complex levels of programming instruction would provide valuable information for researchers, educators, and curriculum designers.

Final remarks

The emergence of new technologies with increasingly sophisticated features has sent researchers on an elusive search for a panacea to the problem of expanding the number and diversity of students entering STEM fields and computer science in particular. The difficulty of programming (Armoni, 2011; Caspersen & Kölling, 2009; Gökçearslan & Alper, 2015; Nilsen & Larsen, 2011; Shadiev et al., 2014) and the use of uninteresting activities in programming instruction (Resnick et al., 2009) are just two of the obstacles facing researchers, curriculum designers, and educators. The use of visual programming languages and robotics have been advocated as promising ways to

introduce novices to programming, but quantitative empirical studies are lacking (Benitti & Barreto, 2012; Poh et al., 2016; Spolaôr & Benitti, 2017). A recent summary of current research exploring the potential of educational robotics in the development of computational thinking (CT) found only nine empirical investigations that intersected the development of CT via the use of educational robotics (Ioannou & Makridou, 2018). Initial evidence supports that educational robotics can foster students' cognitive and social skills, but aspects and challenges remain understudied. They recommended that learning practitioners develop curriculum for computational thinking via robotics and include all of the details about robotics interventions. This study addressed these gaps and recommendations.

In 1980, Papert indicated that "People often ask whether in the future children will program computers or become absorbed in pre-programmed activities" (Papert, 1980, p. 29). This study provided details regarding the instructional tools and techniques that achieved significant improvements in computational thinking and spatial skills as well as competency beliefs, and helped move students one step closer to becoming producers instead of mere consumers of technology. Positive initial student experiences are a necessary first step in the effort to influence future career choices and ultimately expand diversity in STEM.

APPENDICES

APPENDIX A

LESSON PLANS

Appendix A

Lesson Plans

Class period 1 (Scratch and Cozmo): Computational Thinking Pretest

Goals:

1. Students will complete the Computational Thinking Test as a pretest for the unit.

Resources: Copies of Computational Thinking Test

Procedures:

1. Explain the purpose of the pretest and that it will NOT count for a grade.
2. Review instructions for the pretest and ask students to complete it.
3. Collect tests when students finish.

Class period 2: Introduction to coding and code.org

Goals:

1. Students will be introduced to what coders do and learn basic coding with code.org

Resources: Computers with internet access, code.org classroom set up with usernames and passwords ready to go, papers with login information for each student, engagement survey

Procedures:

1. Ask students if they know what coding is and/or what coders do.
2. Show code.org resource video to introduce coding: What most schools don't teach (5 minutes)
3. Demonstrate how to complete a puzzle in code.org
4. Show the code.org resource video: Push yourself. Anybody can learn. (1 minute) Discuss strategies to use if they get stuck and the importance of not giving up.
5. Students log in to code.org
6. Students complete as many puzzles as they can before the end of the period.
7. Students log out before the end of the period.
8. Students complete and submit engagement survey before leaving.

Class period 3 (Scratch): Introduce Scratch

Goals:

1. Students will create a Scratch account.
2. Students will engage in an exploratory, hands-on experience with Scratch.

Resources: Computers with Internet access, Scratch Account Handout, Scratch Surprise Handout with reflection prompts on the back, engagement survey

Procedures:

1. Help students navigate to the Scratch website at http://scratch.mit.edu and click on "Join Scratch" to get started creating a Scratch account. Optionally, have the Scratch Account handout available to guide students. Give students time to register, update their Scratch profile page, and explore the Scratch online community. Encourage students to practice signing in and out of their accounts. Instruct students to write their username and password on their Scratch Account Handout.
2. Instruct students to click on "Create" at the top of the page.
3. Give students 10 minutes to explore the Scratch interface in an open-ended way. Prompt students with, "You have 10 minutes to make something surprising happen to the Scratch cat." Encourage students to work together, ask each other for help, and share what they are figuring out.
4. Ask for 3 or 4 volunteers to share with the entire group one thing that they discovered. Optionally, after the volunteers have shared, offer several challenges to the students:
    a. Did anyone figure out how to add sound?
    b. Did anyone figure out how to change the background?
    c. Did anyone figure out how to get help with blocks?
5. Ask students to respond to the following reflection prompts:
    a. What did you figure out?
    b. What do you want to know more about?
6. Students complete and submit engagement survey before leaving.

Class period 3 (Cozmo): Introduce Cozmo

Goals:

1. Students will introduce themselves to Cozmo.
2. Students will engage in an exploratory, hands-on experience with Cozmo.

Resources: Cozmo robots, cubes, and chargers; iPads; Cozmo Exploration Instructions Handout, reflection prompts, engagement survey

Procedures:

1. Each pair of students should receive a robot, charger, iPad, set of 3 cubes, and Cozmo Explorations Instructions Handout.
2. Raise and lower the lift to reveal the robot's network name and Wi-Fi password.
3. Connect the iPad to the robot's Wi-Fi network. If the tablet says "Internet service not working" it's normal because you're connected to a robot, not the Internet.
4. Run the Cozmo app on the iPad.
5. Let Cozmo run around and see his light cubes, but don't let him fall off the table!
6. Give Cozmo a Tune Up.
7. Feed Cozmo.
8. Play with Cozmo.
9. Give students 10 minutes to explore the Discover button in an open-ended way. Prompt students with, "You have 10 minutes to make something surprising happen with Cozmo." Encourage students to work together, ask each other for help, and share what they are figuring out. Refer them to the Cozmo Explorations Instructions Handout for help/reminders.
10. Ask for 3 or 4 volunteers to share with the entire group one thing that they discovered. Optionally, after the volunteers have shared, offer several challenges to the students:
    a. Did anyone figure out how to meet Cozmo?
    b. Did anyone figure out how to explore the world through Cozmo's eyes?
    c. Did anyone figure out how to get Cozmo to say a word or phrase?
    d. Did anyone figure out how to program Cozmo?
11. Ask students to respond to the following reflection prompts:
    a. What did you figure out?
    b. What do you want to know more about?
12. Students complete and submit engagement survey before leaving.

Class period 4 (Scratch): Step-by-step tutorial

Goals:

1. Students will create a dancing cat by following a step-by-step tutorial.
2. Students will experience building up a program by experimenting and iterating.

Resources: Computers with Internet access, Step-by-step Handout, rubric/critique paper with reflection prompts on the back, engagement survey

Procedures:

1. Help students sign in to their Scratch accounts, distribute the Step-by-Step Handout and click on the Create button at the top of the Scratch website to open the project editor.
2. Have students open the Tips window and follow the Getting Started with Scratch step-by-step tutorial to create a dancing cat program. Encourage students to add

other blocks and experiment with motion, sprites, looks, costumes, sound, or backdrops to make the project their own.

3. Invite students to share their projects with another student.
    a. Students can use a rubric/critique paper to evaluate their own project and then repeat to evaluate their classmate(s) project.

4. Ask students to think back on the design process by responding to the following reflection prompts:
    a. What was surprising about the activity?
    b. How did it feel to be led step-by-step through the activity?
    c. When do you feel most creative?

5. Students complete and submit engagement survey before leaving.

Class period 4 (Cozmo): Step-by-step tutorial

Goals:

1. Students will follow step-by-step tutorials to get Cozmo to perform different sequences of activities.
2. Students will experience building up a program by experimenting and iterating.

Resources: Cozmo robots, cubes, and chargers; iPads; Cozmo blocks library – Sandbox Mode Handout; Cozmo blocks library – Constructor Mode Handout; Constructor Mode New Project Screen Handout; Tutorial Handout; Tutorial – Step by Step Intro Handout; Tutorial – Interactive Steps Handout; Rubric; critique paper with reflection prompts on the back; engagement survey

Procedures:

1. Help students connect to Cozmo and the app.
2. Use the Cozmo blocks library – Sandbox Mode Handout, Cozmo blocks library – Constructor Mode Handout, and Constructor Mode New Project Screen Handouts to demonstrate where to find all of the code block options.
3. Students should tap "Discover" and then "Code Lab" and then "Constructor Mode" and then "New Project."
4. Students will use the tutorial (Tutorial Handout, Tutorial – Step by Step Intro Handout, Tutorial – Interactive Steps Handout) instructions to program Cozmo to perform the sequences described.
5. Students will have the teacher check their code at the 2 specified places.
6. Encourage students to experiment by creating their own codes after they finish the challenges.
7. Invite students to share their projects with another student.
    a. Students can use a rubric/critique paper to evaluate their own project and then repeat to evaluate their classmate(s) project.
8. Ask students to think back on the design process by responding to the following reflection prompts:

a. What was surprising about the activity?
b. How did it feel to be led step-by-step through the activity?
c. When do you feel most creative?
9. Students complete and submit engagement survey before leaving.

Class period 5 (Scratch): 10 block challenge

Goals:

1. Students will create a project with the constraint of only being able to use 10 blocks.

Resources: Computers with Internet access, 10 Blocks Handout, rubric/critique paper with reflection prompts on the back, engagement survey

Procedures:

1. Students will sign in to their Scratch accounts and click on the Create button at the top of the Scratch website to start a new project. Distribute the 10 Blocks Handout to guide students during the activity.
2. Give students time to create a project with only these 10 Scratch blocks: *go to, glide, say, show, hide, set size to, play sound until done, when this sprite clicked, wait,* and *repeat*. Remind students to use each block at least once in their project and encourage them to experiment with different sprites, costumes, and backdrops.
3. Invite students to share their projects with another student/group.
   a. Students can use a rubric/critique paper to evaluate their own project and then repeat to evaluate their classmate(s) project.
4. Ask students to think back on the design process by responding to the following reflection prompts:
   a. What was difficult about being able to use only 10 blocks?
   b. What was easy about being able to use only 10 blocks?
   c. How did it make you think of things differently?
5. Students complete and submit engagement survey before leaving.

Class period 5 (Cozmo): 10 block challenge

Goals:

1. Students will create a project with the constraint of only being able to use 10 blocks.

Resources: Cozmo robots, cubes, and chargers; iPads; 10 Blocks Handout; Cozmo blocks library – Sandbox Mode Handout; Cozmo blocks library – Constructor Mode Handout; Constructor Mode New Project Screen Handout; rubric/critique paper with reflection prompts on the back; engagement survey

Procedures:

1. Help students connect to Cozmo and the app.
2. Students should tap "Discover" and then "Code Lab" and then "Constructor Mode" and then "New Project."
3. Distribute the 10 Blocks Handout and review instructions. Refer students to yesterday's handouts (Cozmo blocks library – Sandbox Mode Handout, Cozmo blocks library – Constructor Mode Handout, Constructor Mode New Project Screen Handout) for help/reminders.
4. Give students time to create a project with only these 10 Cozmo blocks: *Drive, Say, Move lift, Spin lights on cube 1, play animation, play sound, display on Comzo's face, draw line, when cube seen, repeat*. Remind students to use each block at least once in their project and encourage them to experiment.
5. Invite students to share their projects with another student/group.
   a. Students can use a rubric/critique paper to evaluate their own project and then repeat to evaluate their classmate(s) project.
6. Ask students to think back on the design process by responding to the following reflection prompts:
   a. What was difficult about being able to use only 10 blocks?
   b. What was easy about being able to use only 10 blocks?
   c. How did it make you think of things differently?
7. Students complete and submit engagement survey before leaving.

Class period 6 (7 if needed) - (Scratch): Debugging

Goals:

1. Students will investigate the problem and find a solution to five debugging challenges.
2. They will explore a range of concepts (including sequence) through the practices of testing and debugging.
3. They will develop a list of strategies for debugging projects.

Resources: Computers with Internet access, Debug It! Handout, checklist paper with reflection prompts on the back, engagement survey

Procedures:

1. Have the Debug It! Handout available to guide students during the activity.
2. Help students follow the project links listed on the handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
3. Give students time to test and debug each Debug It! Challenge.
4. Ask students to evaluate their success using the checklist and then reflect back on their testing and debugging experiences by responding to the following reflection prompts:

    a. Overall in the lesson, what was the problem you were trying to overcome?
    b. How did you identify the problem in each challenge?
    c. How did you fix the problems?
    d. Did other students have alternative approaches to fixing the problem?
5. Students complete and submit engagement survey before leaving.

Class period 6 (7 if needed) - (Cozmo): Debugging

Goals:

1. Students will investigate the problem and find a solution to five debugging challenges.
2. They will explore a range of concepts (including sequence) through the practices of testing and debugging.
3. They will develop a list of strategies for debugging projects.

Resources: Cozmo robots, cubes, and chargers; iPads; Debug It! Handout, checklist paper with reflection prompts on the back, engagement survey

Procedures:

1. Have the Debug It! Handout available to guide students during the activity.
2. Students should tap "Discover" and then "Code Lab" and then "New Project."
3. Encourage students to investigate the buggy program, tinker with problematic code, and test possible solutions.
4. Give students time to test and debug each Debug It! Challenge.
5. Ask students to evaluate their success using the checklist and then reflect back on their testing and debugging experiences by responding to the following reflection prompts:
    a. Overall in the lesson, what was the problem you were trying to overcome?
    b. How did you identify the problem in each challenge?
    c. How did you fix the problems?
    d. Did other students have alternative approaches to fixing the problem?
6. Students complete and submit engagement survey before leaving.

Class periods 7, 8 - (Scratch): Challenge!

Goals:

1. Students will become familiar with a wider range of blocks.
2. Students will be able to create an open-ended project.

Resources: Computers with Internet Access, About Me Handout, Project Plan Handout, Project Rubric, checklist paper with reflection prompts on the back, engagement survey

Procedures:

1. Distribute the About Me Handout, the Project Plan Handout and the Project Rubric. Explain the instructions and clarify questions.
2. Give students time to create their open-ended projects.
3. Ask students to evaluate their success using the checklist and then reflect back on their testing and debugging experiences by responding to the following reflection prompts:
   a. What are you most proud of? Why?
   b. What did you get stuck on? How did you get unstuck?
   c. What might you want to do next?
   d. What did you discover from looking at others' projects?
4. Students complete and submit engagement survey before leaving.

Class periods 7, 8- (Cozmo): Challenge!

Goals:

1. Students will become familiar with a wider range of blocks.
2. Students will be able to create an open-ended project.

Resources: Cozmo robots, cubes, and chargers; My Project Handout; Project Plan Handout; Project Rubric; checklist paper with reflection prompts on the back; engagement survey

Procedures:

1. Distribute the My Project Handout, the Project Plan Handout, and the Project Rubric. Explain the instructions and clarify questions.
2. Give students time to create their open-ended projects.
3. Ask students to evaluate their success using the checklist and then reflect back on their testing and debugging experiences by responding to the following reflection prompts:
   a. What are you most proud of? Why?
   b. What did you get stuck on? How did you get unstuck?
   c. What might you want to do next?
   d. What did you discover from looking at others' projects?
4. Students complete and submit engagement survey before leaving.

Class period 9 - (Scratch): Share and evaluate Challenge projects

Goals:

1. Students will share and evaluate their Challenge projects.

Resources: Computers with Internet Access, Critique Group Handout with reflection prompts on the back, engagement survey

Procedures:

1. Invite students to share their projects with another group.
2. Give students time to test their classmate(s) projects.
3. Students will use the critique paper to evaluate their classmate(s) project.
   a. Explain that RED is debugging – These are things that don't work or could be improved. They will need to look carefully at the code and watch it run to be sure all of the blocks are functioning.
   b. Explain that YELLOW is something confusing in the code or something that could be done differently.
   c. Explain that GREEN is for things that work well or that you really like.
4. Ask students to turn the critique paper over and respond to the following reflection prompts:
   a. Clarity: Did you understand what the project is supposed to do?
   b. Features: What features does the project have? Does the project work as expected?
   c. Appeal: How engaging is the project? Is it interactive, original, sophisticated, funny, or interesting? How did you feel as you interacted with it?
5. Students will share their feedback on the Critique Group Handout with their classmates.
6. Students will use the feedback to make adjustments to their projects.
7. Students will complete engagement survey.
8. Students will complete the Competency Beliefs Survey.

Class period 9 - (Cozmo): Share and evaluate Challenge projects

Goals:

1. Students will share and evaluate their Challenge projects.

Resources: Cozmo robots, cubes, and chargers; Critique Group Handout with reflection prompts on the back, engagement survey

Procedures:

1. Invite students to share their projects with another group.
2. Give students time to test their classmate(s) projects.
3. Students will use the critique paper to evaluate their classmate(s) project.
   a. Explain that RED is debugging – These are things that don't work or could be improved. They will need to look carefully at the code and watch it run to be sure all of the blocks are functioning.

       b. Explain that YELLOW is something confusing in the code or something that could be done differently.

       c. Explain that GREEN is for things that work well or that you really like.

4. Ask students to turn the critique paper over and respond to the following reflection prompts:

       a. Clarity: Did you understand what the project is supposed to do?

       b. Features: What features does the project have? Does the project work as expected?

       c. Appeal: How engaging is the project? Is it interactive, original, sophisticated, funny, or interesting? How did you feel as you interacted with it?

5. Students will share their feedback on the Critique Group Handout with their classmates.

6. Students will use the feedback to make adjustments to their projects.

7. Students will complete engagement survey.

8. Students will complete the Competency Beliefs Survey.

Class period 10 (Scratch and Cozmo): Wrap Up

Goals:

1. Students will complete the Mental Rotations Test.
2. Students will complete the Computational Thinking Test as a post-test for the unit.
3. Wrap up study.

Resources: Copies of Computational Thinking Test

Procedures:

1. Students will complete the timed Mental Rotations Test.
2. Distribute the post-test and ask students to complete it. Collect when finished.
3. Wrap up study and thank students for participating. Answer any remaining questions that they may have.
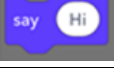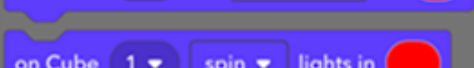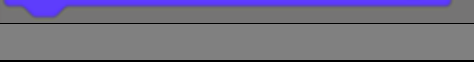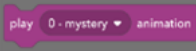
APPENDIX B
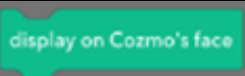
PROJECT RUBRICS

Appendix B

Project Rubrics

Scratch Rubric

Elements to include:	✎ Notes:	☑ Done:
Events		
when 🚩 clicked		
when this sprite clicked		
when space key pressed		
Motion		
move 10 steps		
go to x: 0 y: 0		
glide 1 secs to x: 0 y: 0		
Looks		
say Hello! for 2 secs		
change color effect by 25		
change size by 10		
show		
hide		
Sound		
play sound meow until done		
Control		
wait 1 secs		
repeat 10		
forever		
Project Cohesiveness:		
Clarity (Project does what it is supposed to do)		
Features (Project includes planned features)		
Appeal (Project is engaging)		
Originality (Project is unique)		
Functionality (Project works as expected)		

200

Cozmo Rubric

Elements to include:	✎ Notes:	☑ Done:
Events		
REQUIRED: when 🏳 clicked		
Choose 1: when Cube [any ▼] tapped / when Cube [any ▼] moved / when Cube [any ▼] seen / when face seen / when happy face seen / when sad face seen		
Motion		
REQUIRED: drive 50 mm at 25 mm/s		
Choose 1: turn 90 ° at 45 °/s / drive left wheel -25 mm/s, right 25 mm/s / stop wheels ▼ / navigate 50 forward, -25 to side, 90 ° / navigate in world to 10 x, 10 y, -25 °		
Actions		
REQUIRED: say Hi		
Choose 1: move lift 15 °/s / move lift to 100 % at 45 °/s / move head to 45 ° at 15 °/s / on Cube 1 ▼ set all lights ▼ to 🔴 / on Cube 1 ▼ spin ▼ lights in 🔴		

Display		
REQUIRED: display on Cozmo's face		
Choose 1: draw text at 0 , 31 / set text scale to 100 % / set text alignment to top ▼ left ▼ / draw line from 0 , 0 to 127 , 63 / draw rectangle from 0 , 0 to 127 , 63 / fill rectangle from 0 , 0 to 127 , 63 / draw circle at 63 , 31 with radius 32 / fill circle at 63 , 32 with radius 31		
Animations		
REQUIRED: play 0 - mystery ▼ animation		
Choose 1: play 1 - eighties music ▼ sound / play 1 - eighties music ▼ sound and wait / stop 1 - eighties music ▼ sound		
Control		
REQUIRED: wait 1 secs / repeat 4		
Project Cohesiveness:		
Clarity (Project does what it is supposed to do)		
Features (Project includes planned features)		
Appeal (Project is engaging)		
Originality (Project is unique)		
Functionality (Project works as expected)		

APPENDIX C

STUDENT PROJECT EXAMPLES

Appendix C

Student Project Examples

Scratch Project #1

Scratch Project #2

```
when 🏳 clicked
say Click on the V to learn more about me! for 9 seconds
```

```
when this sprite clicked
say Now you know more about me! for 2 seconds
```

```
when this sprite clicked
move 196 steps
glide 3 secs to x: -160 y: 124
say I dislike coding for 2 seconds
say Press the space key to learn more! for 2 seconds
forever
    play sound Boop Bing Bop until done
```

```
when space key pressed
go to x: 67 y: -23
glide 2 secs to x: -95 y: 128
say My favorite color is blue! for 2 seconds
say Click on the first N to learn more! for 2 seconds
```

```
when this sprite clicked
change ghost effect by 25
say I like to play video games for 2 seconds
wait 1 seconds
say Press the N key to learn more! for 2 seconds
```

```
when n key pressed
hide
go to random position
show
say I really like dragon ball z! for 2 seconds
say Click the 2 key to learn more! for 2 seconds
go to x: 39 y: 137
```

```
when 2 key pressed
repeat 2
    change size by 30
say My favorite animals are dogs! for 2 seconds
say Click the second E to learn more! for 2 seconds
```

```
when this sprite clicked
change size by -30
```

```
when this sprite clicked
glide 1 secs to x: 3 y: 16
say My favorite number is 15! for 1 seconds
go to x: 162 y: 136
say To finish it click on me! for 2 seconds
```

Cozmo Project #1

Cozmo Project #2

REFERENCES

References

ACM, Code.org, CSTA, Cyber Innovation Center, & National Math and Science Initiative. (2016). K-12 Computer Science Framework. Retrieved from https://dl.acm.org/citation.cfm?id=3079760

Aivaloglou, E., & Hermans, F. (2016). How kids code and how we know. *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16*, *September*, 53–61. https://doi.org/10.1145/2960310.2960325

Akbiyik, C. (2009). Can affective computing lead to more effective use of ICT in education? *Revista de Educación*, (352), 179–202. Retrieved from http://www.revistaeducacion.mec.es/re352/re352_08ing.pdf

Akimana, B.-T., Bonnaerens, M., Wilder, J., & Vuylsteker, B. (2017). *A survey of human-tobot interaction in the Internet of things*. Retrieved from https://www.researchgate.net/publication/318722691_A_Survey_of_Human-Robot_Interaction_in_the_Internet_of_Things

Alba, D. (2016). Evo is a little robot with a big mission : Get girls to code. Retrieved from https://www.wired.com/2016/09/ozobot-evo-bot-hopes-get-women-coding/

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society*, *19*(3), 47–57. https://doi.org/https://doi.org/10.2307/jeductechsoci.19.3.47

209

Armoni, M. (2011). The nature of CS in K-12 curricula: The roots of confusion. *ACM Inroads*, *2*(4), 19–20. https://doi.org/10.1145/2038879.2038883

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to "real" programming. *ACM Transactions on Computing Education*, *14*(4), 1–15. https://doi.org/10.1145/2677087

Astleitner, H. (2000). Designing emotionally sound instruction: The FEASP approach. *Instructional Science*, *28*(3), 169–198. https://doi.org/https://doi.org/10.1023/A:1003893915778

Astrid, M. von der P., Krämer, N. C., Gratch, J., & Kang, S.-H. (2010). "It doesn't matter what you are!" Explaining social effects of agents and avatars. *Computers in Human Behavior2*, *26*(6), 1641–1650.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Auerbach, J. E., Concordel, A., Kornatowski, P. M., & Floreano, D. (2018). Inquiry-based learning with RoboGen: An open-source software and hardware platform for robotics and artificial intelligence. *IEEE Transactions on Learning Technologies*, *1382*(c), 1–14. https://doi.org/10.1109/TLT.2018.2833111

Bandura, A. (1977). *Social learning theory*. Englewood Cliffs, NJ: Prentice-Hall.

Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory*. Englewood Cliffs, NJ: Prentice-Hall.

Barker, B. S., Grandgenett, N., Nugent, G., & Adamchuk, V. I. (2010). Robots, GPS/GIS, and programming technologies: The power of "digital manipulatives" in youth extension experiences. *Journal of Extension*, *48*(1), 1–9. Retrieved from http://www.eric.ed.gov/ERICWebPortal/detail?accno=EJ899147

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 Inroads2*, *2*(1), 48–54. https://doi.org/https://doi.org/10.1145/1929887.1929905

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, *20*(1), 45–73.

Benitti, F., & Barreto, V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers and Education*, *58*(3), 978–988. https://doi.org/10.1016/j.compedu.2011.10.006

Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*, *24*(5), 628–647. https://doi.org/10.1007/s10956-015-9552-x

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum.

Computers and Education, *72*, 145–157.

https://doi.org/10.1016/j.compedu.2013.10.020

Blakemore, L. (2017). Does teaching computer programming within Key Stage 1 of the

primary curriculum enhance children ' s problem solving skills? *(Doctoral*

dissertation). University of Sheffield, UK.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrod, P., & Franklin, D. (2013). Hairball:

Lint-inspired static analysis of Scratch projects. In *Proceedings of the 44th ACM*

Technical Symposium on Computer Science Education (pp. 215–220). Denver,

Colorado: ACM. https://doi.org/10.1145/2445196.2445265

Brennan, K., Balch, C., & Chung, M. (2011). An introductory computing curriculum

using Scratch. *Harvard Graduate School of Education*, 154.

https://doi.org/10.1016/S0022-3913(12)00047-9

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the

development of computational thinking. *Annual American Educational Research*

Association Meeting, Vancouver, BC, Canada, 1–25. https://doi.org/10.1.1.296.6602

Brewster, C., & Fager, J. (2000). *Increasing student engagement and motivation: From*

time on task to homework. Portland, OR. Northwest Regional Educational

Laboratory. Retrieved from

https://educationnorthwest.org/sites/default/files/byrequest.pdf

Brichacek, A. (2014). Computational thinking boosts students' higher-order skills.

Retrieved from

https://www.iste.org/explore/articleDetail?articleid=232&category=Featured-

Brown, S. D., & Lent, R. W. (2013). Social cognitive model of career self-management: toward a unifying view of adaptive career behavior across the life span. *Journal of Counseling Psychology*, *60*(4), 557–568. https://doi.org/10.1037/a0033446

Caci, B., Chiazzese, G., & D'Amico, A. (2013). Robotic and virtual world programming labs to stimulate reasoning and visual-spatial abilities. *Procedia - Social and Behavioral Sciences*, *93*, 1493–1497. https://doi.org/10.1016/j.sbspro.2013.10.070

Caci, B., D'Amico, A., & Cardaci, M. (2004). New frontiers for psychology and education: Robotics. *Psyhological Reports*, *94*(3 Pt 2), 1372–1374. https://doi.org/10.2466/pr0.94.3c.1372-1374

Caci, B., D'Amico, A., & Chiazzese, G. (2012). Robotics and virtual worlds: An experiential learning lab. In *Biologically Inspired Cognitive Architectures* (pp. 83–87). Berlin/Heidelberg, Germany: Springer-Verlag.

Carini, R. M., Kuh, G. D., & Klein, S. P. (2006). Student engagement and student learning: Testing the linkages. *Research in Higher Education*, *47*(1), 1–32.

Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (pp. 27–31). Houston, Texas: ACM.

Caspersen, M. E., & Kölling, M. (2009). STREAM: A first programming process. *ACT*

Transaction on Computing Education, *9*(1), 1–29.

https://doi.org/10.1145/1513593.1513597

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics : Using robotics to

motivate math, science, and engineering literacy in elementary school. *International*

Journal of Engineering Education, *22*(4), 711–722.

https://doi.org/10.7771/\n10.7771/2157-9288.1082

Chang, C.-W., Lee, J.-H., Chao, P.-Y., Wang, C.-Y., & Chen, G.-D. (2010). Exploring

the possibility of using humanoid robots as instructional tools for teaching a second

language in primary school. *Educational Technology & Society*, *13*(2), 13–24.

Retrieved from https://www.learntechlib.org/p/75345/

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance

of problem-solving through a visual programming environment. *Computers &*

Education, *95*, 202–215. https://doi.org/10.1016/j.compedu.2016.01.010

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017).

Assessing elementary students' computational thinking in everyday reasoning and

robotics programming. *Computers and Education*, *109*, 162–175.

https://doi.org/10.1016/j.compedu.2017.03.001

Chen, Y.-F., Cannady, M. A., Schunn, C., & Dorph, R. (2017). Measures technical brief:

Competency beliefs in STEM. Retrieved from http://activationlab.org/wp-

content/uploads/2018/03/CompetencyBeliefs_STEM-Report_20170403.pdf

Cheng, Y. W., Sun, P. C., & Chen, N. S. (2018). The essential applications of educational robot: Requirement analysis from the perspectives of experts, researchers and instructors. *Computers and Education, 126*(1), 399-416. https://doi.org/10.1016/j.compedu.2018.07.020

Chin, K.-Y., Hong, Z.-W., & Chen, Y.-L. (2014). Impact of using an educational robot-based learning system on students' motivation in elementary education. *IEEE Transactions on Learning Technologies*, *7*(4), 333–345. https://doi.org/10.1109/TLT.2014.2346756

Chung, J., Cannady, M. A., Schunn, C., Dorph, R., & Bathgate, M. (2016). Measures technical brief: Engagement in science learning activities. Retrieved from http://activationlab.org/wp-content/uploads/2018/03/Engagement-Report-3.2-20160803.pdf

Clark, D. B., Tanner-Smith, E. E., & Killingsworth, S. S. (2015). Digital games, design, and learning. *Review of Educational Research*, *86*(1), 79–122. https://doi.org/10.3102/0034654315582065

CollegeBoard. (2017). College Board AP data sets. Retrieved from https://research.collegeboard.org/programs/ap/data/participation/ap-2017

Coxon, S. V. (2012). The malleability of spatial ability under treatment of a FIRST LEGO League-based robotics simulation. *Journal for the Education of the Gifted*, *35*(3), 292–316. https://doi.org/10.II77/01623S3212451788

CSTA. (2011). CSTA K-12 computer science standards. Retrieved from

https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/CST

A_K-12_CSS.pdf

D'Amico, A., & Guastella, D. (2018). Robotics construction kits: From "objects to think

with" to "objects to think and to emote with." *Future Internet*, *10*(2), 1–6.

https://doi.org/10.3390/fi10020021

Dagiene, V., & Stupurienė, G. (2015). Bebras - a sustainable community building model

for the concept based learning of informatics and computational thinking.

Informatics in Education, *15*(1), 25–44.

Demir, K. A. (2017). Research questions in roboethics. *Mugla Journal of Science and

Technology*, *3*(2), 160–165. https://doi.org/10.22531/muglajsci.359648

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 and Education, *58*(1), 240–249.

https://doi.org/10.1016/j.compedu.2011.08.006

Denning, P. J. (2017). Remaining trouble spots with computational thinking.

Communications of the ACM, *60*(6), 33–39. https://doi.org/10.1145/2998438

Dettori, G., Bocconi, S., Chioccariello, A., Ferrari, A., Engelhardt, K., Kampylis, P., &

Punie, Y. (2016). Developing computational thinking: Approaches and orientations

in K-12 education. In *EdMedia 2016* (pp. 1–7). Vancouver, British Columbia.

https://doi.org/https://doi.org/10.2791/792158

DfE. (2013). The national curriculum in England: Framework document. Retrieved from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachm ent_data/file/381344/Master_final_national_curriculum_28_Nov.pdf

Dierking, L. D., Falk, J. H., Rennie, L., Anderson, D., & Ellenbogen, K. (2003). Policy statement of the "informal science education" ad hoc committee. *Journal of Research in Science Teaching*, *40*(2), 108–111. https://doi.org/https://doi.org/10.1002/tea.10066

DiSessa, A. (2001). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

Diwanji, P. (2017). Anki's new coding app uses Scratch Blocks to help anyone program their Cozmo robot. Retrieved from https://www.blog.google/topics/education/ankis-new-coding-app-uses-scratch-blocks-help-anyone-program-their-cozmo-robot/

Djambong, T., & Freiman, V. (2016). Task-based assessment of students' computational thinking skills developed through visual programming or tangible. In *13th International Conference on Cogition and Exploratory Learning in Digital Age (CELDA 2016)* (pp. 41–51).

Dolgopolovas, V., Jevsikova, T., Dagiene, V., & Savulioniene, L. (2016). Exploration of computational thinking of software engineering novice students based on solving computer science tasks. *International Journal of Engineering Education*, *32*(3(A)),

1–10.

Donnelly, M. (2013). STEM fields and STEM education. *Research Starters*, *Education*.

Druga, S., Williams, R., Breazeal, C., & Resnick, M. (2017). "Hey Google is it OK if I eat you?" *Proceedings of the 2017 Conference on Interaction Design and Children - IDC '17*, 595–600. https://doi.org/10.1145/3078072.3084330

Edwards, A., Edwards, C., Spence, P. R., Harris, C., & Gambino, A. (2016). Robots in the classroom: Differences in students' perceptions of credibility and learning between "teacher as robot" and "robot as teacher." *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2016.06.005

Eguchi, A. (2009). What are students learning from educational robotics? Different approaches to educational robotics. In *Association for the Advancement of Computing Education (AACE) Society for Information Technology & Teacher Education International Conference* (pp. 3547–3554). Retrieved from https://www.learntechlib.org/p/31201/

Fasola, J., & Mataric, M. (2013). A socially assistive robot exercise coach for the elderly. *Journal of Human-Robot Interaction*, *2*(2), 3–32.

Fayer, S., Lacey, A., & Watson, A. (2017). STEM occupations: Past, present and future. Retrieved from https://www.bls.gov/spotlight/2017/science-technology-engineering-and-mathematics-stem-occupations-past-present-and-future/pdf/science-technology-engineering-and-mathematics-stem-occupations-past-present-and-future.pdf.

Fernández-Llamas, C., Conde, M. A., Rodríguez-Lera, F. J., Rodríguez-Sedano, F. J., & García, F. (2018). May I teach you? Students' behavior when lectured by robotic vs. human teachers. *Computers in Human Behavior, 80,* 460-469. https://doi.org/10.1016/j.chb.2017.09.028

Fesakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016.

Fesakis, G., & Serafeim, K. (2009). Influence of the familiarization with " Scratch " on future teachers' opinions and attitudes about programming and ICT in education. *ACM SIGCSE Bulletin*, *41*(3), 258–262. https://doi.org/10.1145/1595496.1562957

Field, A. (2013). *Discovering statistics using IBM SPSS statistics: and sex and drugs and rock "n" roll* (4th ed.). Los Angeles: Sage.

Fields, D. A., Kafai, Y. B., & Giang, M. T. (2017). Youth computational participation in the wild. *ACM Transactions on Computing Education*, *17*(3), 1–22. https://doi.org/10.1145/3123815

Finn, J. D., Pannozzo, G. M., & Voelkl, K. E. (1995). Disruptive and inattentive-withdrawn behavior and achievement among fourth graders. *The Elementary School Journal*, 421–434.

Fortus, D., Krajcik, J., Dershimer, R. C., Marx, R. W., & Mamlok-Naaman, R. (2005). Design-based science and real-world problem-solving. *International Journal of*

Science Education, *27*(7), 855–879. https://doi.org/10.1080/09500690500038165

Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research*, *74*(1), 59–109. https://doi.org/https://doi.org/10.3102/00346543074001059

Fredricks, J. A., McColskey, W., Meli, J., Mordica, J., Montrosse, B., & Mooney, K. (2011). Measuring student engagement in upper elementary through high school: A description of 21 instruments. *Issues & Answers* (Vol. REL 2011-N).

Freeman, A., Becker, S., Cummins, M., Davis, A., & Hall Giesinger, C. (2017). *NMC/CoSN Horizon report: 2017 K-12 edition*. Austin, TX. Retrieved from https://www.epiphanymgmt.com/Downloads/horizon report.pdf

Fronza, I., El Ioini, N., & Corral, L. (2017). Leveraging robot programming to foster computational thinking. *Proceedings of the 9th International Conference on Computer Supported Education*, *2*(Csedu), 109–116. https://doi.org/10.5220/0006310101090116

Gaudiello, I., & Zibetti, E. (2016). *Learning robotics, with robotics, by robotics: Educational robotics*, *volume 3*. London, UK: ISTE Ltd. https://doi.org/10.1002/9781119335740

Giles, J. (2017). A class for the masses. Retrieved from http://www.news.gatech.edu/features/class-notes-class-masses

Gökçearslan, Ş., & Alper, A. (2015). The effect of locus of control on learners ' sense of

community and academic success in the context of online learning commununities. *The Internet and Higher Education*, *27*, 64–73. https://doi.org/10.1016/j.iheduc.2015.06.003

Gökçearslan, Ş., Günbatar, M. S., & Kukul, V. (2017). Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation and reliability. *Eğitim Teknolojisi Kuram ve Uygulama*, *7*(1), 158–158. https://doi.org/10.17943/etku.288493

Gorman, C., & Ackerman, E. (2017). Anki's Code Lab brings sophisticated graphical programming to Cozmo robot. Retrieved from https://spectrum.ieee.org/automaton/robotics/diy/anki-code-lab-brings-sophisticated-graphical-programming-to-cozmo-robot

Grover, S. (2014). *Foundations for advancing computational thinking: Balanced designs for deeper learning in a computer science course for middle school students (Doctoral dissertation)*. Stanford University, CA.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Grover, Shuchi, Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education - ITiCSE '14*, 57–62. https://doi.org/10.1145/2591708.2591713

Grover, Shuchi, 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. https://doi.org/10.1080/08993408.2015.1033142

Grubbs, M. (2013). Robotics intrigue middle school students and build STEM skills. *Technology and Engineering Teacher*, *72*(6), 12–16. Retrieved from https://www.learntechlib.org/p/131717/

Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, *8*(6), 1–165. https://doi.org/https://doi.org/10.2200/S00684ED1V01Y201511HCI033

Guzdial, M., & Forte, A. (2005). Design process for a non-majors computing course. *ACM SIGCSE Bulletin*, *37*(1), 361–365. https://doi.org/10.1145/1047124.1047468

Hegarty, M., & Waller, D. A. (2005). Individual differences in spatial abilities. In P. Shah & A. Miyake (Ed.), *The Cambridge Handbook of Visuospatial Thinking* (pp. 121–169). New York, NY: Cambridge University Press.

Heintz, F., Mannila, L., & Tommy, F. (2016). A review of models for introducing computational thinking , computer science and computing in K – 12 education. *Proceedings - Frontiers in Education Conference, FIE*, *2016* (November), 1–9. Retrieved from https://doi.org/10.1109/FIE.2016.7757410

Henderson, P. B., Cortina, T. J., Hazzan, O., & Wing, J. M. (2007). Computational thinking. *Communications of the Association for Computing Machinery (ACM)*,

49(3), 33–35. https://doi.org/https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

Hermans, C. (2017). 15-494/694: Cognitive robotics. Retrieved from

https://www.cs.cmu.edu/afs/cs/academic/class/15494-s17/

Hermans, F., & Aivaloglou, E. (2016). Do code smells hamper novice programming: A

controlled experiment on Scratch programs. In *Proceedings of the 24th International

Conference on Program Comprehension* (pp. 1–10). Danvers, MA.

https://doi.org/10.1109/ICPC.2016.7503706

Hewner, M. (2013). Undergraduate conceptions of the field of computer science. In

*Proceedings of the ninth annual international ACM conference on International

computing education research* (pp. 107–114). San Diego, CA: ACM.

https://doi.org/10.1145/2493394.2493414

Highfield, K. (2010). Robotic toys as a catalyst for mathematical problem solving.

Australian Mathematics Teacher, *15*(2), 22–27. Retrieved from

https://www.researchonline.mq.edu.au/vital/access/services/Download/mq:12742/D

S01?view=true

Hinton, T. B. (2018). An exploratory study of a robotics educational platform on STEM

career interests in middle school students. *Dissertation Abstracts International*, *78*,

146.

Huang, Q., Yun, T., Yujin, W., Changlin, W., & Lianqing, Y. (2018). A review of

cognitive psychology applied in robotics. *Advances in Intelligent Systems and*

Interactive Applications, *686*, 125–133. https://doi.org/10.1007/978-3-319-69096-4

Hussain, S., Lindh, J., & Shukur, G. (2006). The effect of LEGO training on pupils' school performance in mathematics, problem solving ability and attitude: Swedish data. *Educational Technology & Society*, *9*(3), 182–194. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.4677&rep=rep1&type =pdf

Ioannou, A., & Makridou, E. (2018). Exploring the potentials of educational robotics in the development of computational thinking : A summary of current research and practical proposal for future work. *Education and Information Technologies*, *23*(6), 2531–2544. https://doi.org/https://doi.org/10.1007/s10639-018-9729-z

Jablon, J. R., & Wilkinson, M. (2006). Using engagement strategies to facilitate children's learning and success. *Beyond the Journal, Young Children on the Web*, 1–5. Retrieved from https://www.naeyc.org/files/yc/file/200603/JablonBTJ.pdf

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology*, *26*(2), 175–192. https://doi.org/10.1007/s10956-016-9663-z

Julià, C., & Antolí, J. Ò. (2016). Spatial ability learning through educational robotics. *International Journal of Technology and Design Education*, *26*(2), 185–203. https://doi.org/10.1007/s10798-015-9307-2

Kaiser. (2018). From STEM to learn. Retrieved from

https://developer.anki.com/blog/features/interview/from-stem-to-learn/

Kalelioglu, F. (2015). A new way of teaching programming skills to K-12 students:

Code.org. *Computers in Human Behavior*, *52*, 200–210.

https://doi.org/10.1016/j.chb.2015.05.047

Kandlhofer, M., & Steinbauer, G. (2016). Evaluating the impact of educational robotics

on pupils' technical- and social-skills and science related attitudes. *Robotics and*

Autonomous Systems, *75*, 679–685. https://doi.org/10.1016/j.robot.2015.09.007

Karim, M. E., Lemaignan, S., & Mondada, F. (2016). A review: Can robots reshape K-12

STEM education? *Proceedings of IEEE Workshop on Advanced Robotics and Its*

Social Impacts, ARSO, *2016-March*, 1–8.

https://doi.org/10.1109/ARSO.2015.7428217

Kay, R. H., & Knaack, L. (2005). A case for ubiquitous, integrated computing in teacher

education. *Technology, Pedagogy and Education*, *14*(3), 391–412.

https://doi.org/10.1080/14759390500200213

Kazakoff, E., & Bers, M. U. (2012). Programming in a robotics context in the

kindergarten classroom: The impact on sequencing skills. *Journal of Educational*

Multimedia and Hypermedia, *21*, 371–391.

Ke, F. (2014). An implementationa of design-based learning through creating educational

computer games: A case study on mathematics learning during design and

computing. *Computers & Education*, *73*(1), 26–39.
https://doi.org/10.1016/j.compedu.2013.12.010

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, *37*(2), 83–137.

Keller, J. M. (1987). Development and use of the ARCS model of instructional design. *Journal of Instructional Development*, *10*(3), 2–10.
https://doi.org/https://doi.org/10.1007/BF02905780

Keller, J. M. (2010). *Motivational design for learning and performance: The ARCS model approach*. Boston, MA: Springer US.
https://doi.org/https://doi.org/10.1007/978-1-4419-1250-3

Kennington, C., & Plane, S. (2017). Symbol, conversational, and societal grounding with a toy robot. *Association for the Advancement of Artificial Intelligence*, 2016–2018. Retrieved from http://arxiv.org/abs/1709.10486

Keren, G., & Fridin, M. (2014). Kindergarten Social Assistive Robot (KindSAR) for children's geometric thinking and metacognitive development in preschool education: A pilot study. *Computer in Human Behavior*, *35*, 400–412.
https://doi.org/http://dx.doi.org/10.1016/j.chb.2014.03.009

Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning,

and teaching. *Computers and Education*, *91*, 14–31.
https://doi.org/10.1016/j.compedu.2015.08.005

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, *20*(3), 362–404. https://doi.org/10.1016/0010-0285(88)90004-7

Klassner, F., & Anderson, S. D. (2003). LEGO Mindstorms: Not just for K-12 anymore. *IEEE Robotics & Automation Magazine*, *10*(2), 12–18. https://doi.org/10.1109/MRA.2003.1213611

Kleinschmager, S., & Hanenberg, S. (2011). How to rate programming skills in programming experiments? A preliminary, exploratory study based on university marks. *Proceedings of the 3rd ACM SIGPLAN Workshop on Evaluation and Usability of Programming Languages and Tools*, 15–24. https://doi.org/10.1145/2089155.2089161

Korkmaz, Özgen, Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scale (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

Kurland, D. M., & Pea, R. D. (1985). Children's mental models of recursive LOGO programs. *Journal of Educational Computing Research*, *1*(2), 235–243. https://doi.org/10.2190/JV9Y-5PD0-MX22-9J4Y

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the

development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, *2*(4), 429–458. https://doi.org/10.2190/BKML-B1QV-KDN4-8ULH

Kwak, S. S., Kim, Y., Kim, E., Shin, C., & Cho, K. (2013). What makes people empathize with an emotional robot? The impact of agency and physical embodiment on human empathy for a robot. *2013 IEEE ROMAN Proceedings of IEEE International Symposium of Robot Human Interactive Communication, 22nd*, *22nd* (Gyeongju, Korea), 180–185. https://doi.org/10.1109/ROMAN.2013.6628441

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14–18. https://doi.org/10.1145/1151954.1067453

Lauwers, T., Nourbakhsh, I. R., & Hamner, E. (2009). CS bots: Design and deployment of a robot designed for the CS1 classroom. *ACM SIGCSE Bulletin*, *41*(1), 428–432.

Lawson, A. E., Banks, D. L., & Logvin, M. (2007). Self-efficacy, reasoning ability, and achievement in college biology. *Journal of Research in Science Teaching*, *44*(5), 706–724. https://doi.org/https://doi.org/10.1002/tea.20172

Lee, H. R., & Sabanović, S. (2014). Culturally variable preferences for robot design and use in South Korea, Turkey, and the United States. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 17–24). Bielefeld, Germany. https://doi.org/10.1145/2559636.2559676

Leonard, J., Buss, A., Gamboa, R., Mitchell, M., Fashola, O. S., Hubert, T., & Almughyirah, S. (2016). Using robotics and game design to enhance children's self-efficacy, STEM attitudes, and computational thinking skills. *Journal of Science Education and Technology*, *25*(6), 860–876. https://doi.org/10.1007/s10956-016-9628-2

Leonard, J., Unertl, A., Barnes-johnson, J., Stubbe, C. R., Mitchell, M., & Ingraham, L. (2017). Developing teachers' computational thinking beliefs and engineering practices through game design and robotics. In *Proceedings of the 39th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education* (pp. 1289–1296). Indianapolis: Association of Mathematics Teacher Educators.

Levy, R. B., & Ben-Ari, M. M. (2015). Robotics activities - Is the investment worthwhile. *ISSEP 2015*, *9378*, 22–31. https://doi.org/10.1007/978-3-319-25396-1

Lewis, C. M. (2010). How programming environment shapes perception, learning and goals. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education - SIGCSE '10*, 346–350. https://doi.org/10.1145/1734263.1734383

Li, J. (2015). The benefit of being physically present: A survey of experimental works comparing copresent robots, telepresent robots, and virtual agents. *International Journal of Human-Computer Studies*, *77*, 23–37.

Lindh, J., & Holgersson, T. (2007). Does LEGO training stimulate pupils' ability to solve logical problems? *Computers & Education*, *49*(4), 1097–1111.

https://doi.org/10.1016/j.compedu.2005.12.008

Linnenbrink, E. A., & Pintrich, P. R. (2003). The role of self-efficacy beliefs in student engagement and learning in the classroom. *Reading & Writing Quarterly: Overcoming Learning Difficulties*, *19*(2), 119–137. https://doi.org/http://dx.doi.org/10.1080/10573560308223

Liu, A. S., Schunn, C. D., Flot, J., & Shoop, R. (2013). The role of physicality in rich programming environments. *Computer Science Education*, *23*(4), 315–331. https://doi.org/10.1080/08993408.2013.847165

Liu, D.-J., Huang, R.-H., Chen, N.-S., & Fan, L. (2016). *The current situation and trend of global development of educational robots.* Beijing: Posts & Telecommunications Press.

Liu, L. L., Uttal, D. H., Marulis, L. M., & Newcombe, N. S. (2008). Training spatial skills: What works, for whom, why and for how long! In *20th annual meeting of the Association for Pyschological Science*. Chicago, IL.

Lonati, V., Malchiodi, D., Monga, M., & Morpurgo, A. (2015). Is coding the way to go? *ISSEP 2015*, *9378*, 165–174. https://doi.org/10.1007/978-3-319-25396-1

Loyd, B. H., & Gressard, C. (1984). Reliability and factorial validity of CAS. *Journal of Educational and Psychological Measurement*. https://doi.org/10.1177/0013164484442033

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. https://doi.org/10.1016/j.chb.2014.09.012

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, *39*(1), 223. https://doi.org/10.1145/1227504.1227388

Malcom, S., Teich, A. H., Jesse, J. K., Campbell, L. A., Babco, E. L., & Bell, N. E. (2005). Preparing women and minorities for the IT workforce: The role of nontraditional educational pathways. *Report by AAAS Education and Human Resources Programs, AAAS Science & Policy Programs, and the Commission on Professionals in Science and Technology*, 1–190. Retrieved from file:///V:/ARC/Lit Review Database/Articles/AAAS women minorities report.pdf

Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *SIGCSE '08 Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 367–371. https://doi.org/10.1145/1352135.1352260

Manches, A., & Plowman, L. (2017). Computing education in children's early years: A call for debate. *British Journal of Educational Technology*, *48*(1), 191–201. https://doi.org/10.1111/bjet.12355

Marginson, S., Tytler, R., Freeman, B., & Roberts, K. (2013). *STEM: country comparisons: International comparisons of science, technology, engineering and mathematics (STEM) education. Final report. Australian Council of Learned Academies*. Melbourne, Vic. https://doi.org/ISBN 978 0 9875798 0 5

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). *Stuck in the shallow end: Education, race and computing*. Cambridge, MA: MIT Press.

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.

Markham, S. A., & King, K. N. (2010). Using personal robots in CS1: Experiences, outcomes, and attitudinal influences. In *15th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 204–208). Bikent, Ankara, Turkey: ACM. https://doi.org/978-1-60558-820-9/10/06

Marks, H. M. (2000). Student engagement in instructional activity: Patterns in the elementary, middle and high school years. *American Educational Research Journal*, *37*(1), 153–184. Retrieved from http://www.jstor.org/stable/1163475%0D

Marks, H. M. (2013). Student engagement in instructional activity: Patterns in the elementary, middle, and high school years. *American Educational Research Journal*, *37*(1), 153–184.

Martines, J. (2018). The future is here : Montour middle schoolers to start AI curriculum this fall. Retrieved from https://archive.triblive.com/news/education-classroom/the-future-is-here-montour-middle-schoolers-to-start-ai-curriculum-this-fall/

McGill, M. M. (2012). Learning to program with personal robots: Influences on student motivation. *ACM Transactions on Computing Education*, *12*(1), Article 4. https://doi.org/10.1145/2133797.2133801

McPherson, S. (2014). Strategies and resources for preparing teachers for STEM teaching and learning. *Proceedings of Society for Information Technology & Teacher Education International Conference*, 1927–1939. Retrieved from https://www.learntechlib.org/primary/p/131068/

Merino-Armero, J. M., González-Calero, J. A., Cózar-Gutiérrez, R., & Villena-Taranilla, R. (2018). Computational thinking initiation. An experience with robots in primary education. *Journal of Research in Science Mathematics and Technology Education*, *1*(2), 181–206. https://doi.org/10.31756/jrsmte.124

Merisuo-Storm, T. (2006). Girls and boys like to read and write different texts. *Scandinavian Journal of Educational Research*, *50*(2), 111–125. https://doi.org/10.1080/00313830600576039

Mitnik, R., Nussbaum, M., & Recabarren, M. (2009). Developing cognition with collaborative robotic activities. *Educational Technology & Society*, *12*(4), 317–330. Retrieved from https://www.j-ets.net/ETS/journals/12_4/27.pdf

Mitnik, R., Recabarren, M., Nussbaum, M., & Soto, A. (2009). Collaborative robotic instruction: A graph teaching experience. *Computers & Education*, *53*(2), 330–342. Retrieved from https://www.learntechlib.org/p/67051/

Monroy-Hernández, A., & Resnick, M. (2015). Empowering kids to create and share programmable media. *Interactions*, *15*(2), 50–53. https://doi.org/10.1145/1340961.1340974

Moreno-León, J., & Robles, G. (2014). Automatic detection of bad programming habits in Scratch: A preliminary study. In *IEEE Frontiers in Education* (pp. 1–4).

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of Scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, *46*, 1–23. Retrieved from http://www.um.es/ead/red/46/moreno_robles.pdf

Moreno-León, J., Robles, G., & Román-González, M. (2016). Comparing computational thinking development assessment scores with software complexity metrics. In *IEEE Global Engineering Education Conference* (pp. 2–10). Abu Dhabi, United Arab Emirates. https://doi.org/http://dx.doi.org/10.1109/EDUCON.2016.7474681

Mubin, O., Stevens, C. J., Shahid, S., Mahmud, A. Al, & Dong, J.-J. (2013). A review of the applicability of robots in education. *Technology for Education and Learning*, *1*(1), 1–7. https://doi.org/10.2316/Journal.209.2013.1.209-0015

NGSS. (2017). *Next generation science standards*. Retrieved from https://dl.acm.org/citation.cfm?id=3079760

Nikou, S. A., & Economides, A. A. (2014). Transition in student motivation during a Scratch and an App Inventor course. *IEEE Global Engineering Education Conference, EDUCON*, (April), 1042–1045. https://doi.org/10.1109/EDUCON.2014.6826234

Nilsen, H., & Larsen, A. (2011). Using the personalized system of instruction in an

introductory programming course. *NOKOBIT, November 2*, 27–38.

Nourbakhsh, I. R., Hamner, E., Crowley, K., & Wilkinson, K. (2004). Formal measures of learning in a secondary school mobile robotics course. In *IEEE International Conference on Robotics and Automation*. New Orleans, LA. https://doi.org/10.1109/ROBOT.2004.1308090

NRC, N. R. C. (2006). *Learning to think spatially*. Washington, DC: The National Academies Press. https://doi.org/https://doi.org/10.17226/11019

Nugent, G., Barker, B., Grandgenett, N., & Adamchuk, V. I. (2010). Impact of robotics and geospatial technology interventions on youth STEM learning and attitudes. *Journal of Research on Technology in Education*, *42*(4), 391–408. https://doi.org/10.1080/15391523.2010.10782557

Nugent, G., Barker, B., Welch, G., Grandgenett, N., Wu, C. R., & Nelson, C. (2015). A model of factors contributing to STEM learning and career orientation. *International Journal of Science Education*, *37*(7), 1067–1088. https://doi.org/10.1080/09500693.2015.1017863

Ormrod, J. E. (2012). *Human learning* (6th ed.). Upper Saddle River, NJ: Pearson Education Inc.

Ospennikova, E., Ershov, M., & Iljin, I. (2015). Educational robotics as an inovative educational technology. *Procedia - Social and Behavioral Sciences, 214,* 18-26. https://doi.org/10.1016/j.sbspro.2015.11.588

Özüorçun, N. Ç., & Bicen, H. (2017). Does the inclusion of robots affect engineering students' achievement in computer programming courses? *Eurasia Journal of Mathematics, Science and Technology Education*, *13*(8), 4779–4787. https://doi.org/10.12973/eurasia.2017.00964a

Paauwe, R. A., Hoorn, J. F., Konijn, E. A., & Keyson, D. V. (2015). Designing robot embodiments for social interaction: Affordances topple realism and aesthetics. *International Journal of Social Robotics*, *7*(5), 697–708.

Pajares, F. (1996). Self-efficacy beliefs in academic settings. *Review of Educational Research*, *66*(4), 543–578. https://doi.org/http://doi.org/10.3102/00346543066004543

Pantic, M., Evers, V., Deisenroth, M., Merino, L., & Schuller, B. (2016). Social and affective robotics tutorial. *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*, 1477–1478. https://doi.org/10.1145/2964284.2986914

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016). Using Scratch and App Inventor for teaching introductory programming in secondary education. A case study. *International Journal of Technology Enhanced Learning*, *8*(3/4), 217. https://doi.org/10.1504/IJTEL.2016.082317

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2017). Empirical studies on the Maker Movement, a promising approach to learning: A literature review. *Entertainment Computing*, *18*, 57–78. https://doi.org/10.1016/j.entcom.2016.09.002

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. New York, NY: Basic Books. https://doi.org/10.1016/0732-118X(83)90034-X

Papert, S., & Harel, I. (1991). Situating constructionism. In Papert & Harel, *Constructionism* (Chapter 1). Norwood, NJ: Ablex Publishing Corporation. Retrieved from http://web.media.mit.edu/~calla/web_comunidad/Reading-En/situating_constructionism.pdf

Pea, R. (1983). Logo programming and problem solving. In *Symposium of the American Educational Research Association* (pp. 1–10). Montreal, Canada.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137–168. https://doi.org/10.1016/0732-118X(84)90018-7

Peters, M., Laeng, B., Latham, K., Jackson, M., Zaiyouna, R., & Richardson, C. (1995). A redrawn Vandenberg & Kuse Mental Rotations Test: Different versions and factors that affect performance. *Brain and Cognition*, *28*(1), 39–58.

Petre, M., & Price, B. (2004). Using robotics to motivate 'Back Door' learning. *Education and Information Technologies*, *9*(2), 147–158.

Phetsrikran, T., Massagram, W., & Harfield, A. (2017). First steps in teaching computational thinking through mobile technology and robotics. *Asian International Journal of Social Sciences*, *17*(3), 37–52. https://doi.org/https://doi.org/10.29139/aijss.20170303

Picard, R. W. (2003). Affective computing: Challenges. *International Journal of Human-Computer Studies, 59*, 55–64. https://doi.org/10.1016/S1071-5819(03)00052-1

Picard, R. W., Vyzas, E., & Healey, J. (2001). Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(10), 1175–1191. https://doi.org/10.1109/34.954607

Plass, J. L., Heidig, S., Hayward, E. O., Homer, B. D., & Um, E. (2013). Emotional design in multimedia learning: Effects of shape and color on affect and learning. *Learning and Instruction*, *29*, 128–140. https://doi.org/10.1016/j.learninstruc.2013.02.006

Poh, L., Toh, E., Causo, A., Tzuo, P.-W., Chen, I.-M., & Yeo, S. H. (2016). A review on the use of robots in education and young children. *Educational Technology & Society*, *19*(2), 148–163. https://doi.org/10.2307/jeductechsoci.19.2.148

Price, J. (2010). The effect of instructor race and gender on student persistence in STEM fields. *Economics of Education Review*, *29*(6), 901–910. https://doi.org/10.1016/j.econedurev.2010.07.009

Price, T. W., & Barnes, T. (2015). Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 91–99). New York, NY: ACM.

Pugnali, A., & Sullivan, A. (2017). The impact of user interface on young children's

computational thinking. *Journal of Information Technology Education: Innovations in Practice*, *16*(16), 171–193. Retrieved from http://www.informingscience.org/Publications/3768

Resnick, M. (2013). Learn to code, code to learn. *EdSurge*, *May 2013*. Retrieved from https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*, 60–67. https://doi.org/10.1145/1592761.1592779

Robinson, M. (2005). Robotics-driven activities: Can they improve middle school science learning? *Bulletin of Science, Technology & Society*, *25*(1), 73–84. https://doi.org/10.1177/0270467604271244

Rogers, C., & Portsmore, M. (2004). Bringing engineering to elementary school. *Journal of STEM Education*, *5*(3), 17–29.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A

two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129–141. https://doi.org/https://doi.org/10.1016/j.compedu.2016.03.003

Salvini, P., Korsah, A., & Nourbakhsh, I. (2016). Yet another robot application? *IEEE Robotics & Automation Magazine*, *23*(2), 12–15. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7494833

Sandars, M., Van Oss, T., & McGeary, S. (2016). ISTE standards for students. *Journal of Experiential Education*, *39*(1), 73–88.

Sarmento, H. R., Reis, C. A. S., Zaramella, V., Almeida, L. D. A., & Tacla, C. A. (2015). Supporting the development of computational thinking: A robotic platform controlled by smartphone. In: P. Zaphiris & A. Ioannou (Eds.), *Learning and Collaboration Technologies*, *9192* (Lecture Notes in Computer Science), 124–135. https://doi.org/10.1007/978-3-319-20609-7_13

Scherer, K. R. (2005). What are emotions? And how can they be measured? *Social Science Information*, *44*(4), 695–729. https://doi.org/https://doi.org/10.1177/0539018405058216

Schmidt, E., & Cohen, J. (2013). *The new digitial age: Transforming nations, businesses, and our lives*. New York, NY: Knopf Doubleday Publishing Group.

Schunk, D. H., Pintrich, P. R., & Meece, J. L. (2008). *Motivation in education: Theory, research, and applications* (3rd ed.). Upper Saddle River, NJ: Pearson Education Inc.

Schutz, P. A., & DeCuir, J. T. (2002). Inquiry on emotions in education. *Educational Psychologist*, *37*(2), 125–134. https://doi.org/10.1207/S15326985EP3702_7

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research - ICER '13*, 59–66. https://doi.org/10.1145/2493394.2493403

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469–495. https://doi.org/10.1007/s10639-016-9482-0

Shadiev, R., Hwang, W.-Y., Yeh, S.-C., Yang, S. J. H., Wang, J.-L., Han, L., & Hsu, G.-L. (2014). Effects of unidirectional vs. reciprocal teaching strategies on web-based computer programming learning. *Journal of Educational Computing Research*, *50*(1), 67–95. https://doi.org/10.2190/EC.50.1.d

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Siegler, R. S. (1986). Piaget's theory of development. In *Children's thinking* (pp. 21–61). Englewood Cliffs, NJ: Prentice-Hall.

Silk, E., Schunn, C., & Shoop, R. (2009). Synchronized robot dancing: Motivating efficiency & meaning in problem-solving with robotics. *Robot Magazine*, (17), 74–

77. Retrieved from https://www.ri.cmu.edu/publications/synchronized-robot-dancing-motivating-efficiency-meaning-in-problem-solving-with-robotics/

Sims, V. K., & Mayer, R. E. (2002). Domain specifity of spatial expertise: The case of video game players. *Applied Cognitive Psychology*, *16*(1), 97–115. https://doi.org/http://dx.doi.org/10.1002/acp.759

Spolaôr, N., & Benitti, F. B. V. (2017). Robotics applications grounded in learning theories on tertiary education: A systematic review. *Computers & Education, 112,* 97-107 . https://doi.org/10.1016/j.compedu.2017.05.001

Stager, G. S. (2005). Papertian constructionism and the design of productive contexts for learning. In *EuroLogo* (pp. 1–11). Warsaw, Poland. Retrieved from http://www.stager.org/articles/eurologo2005.pdf

Statt, N. (2016). Anki's Cozmo robot is the real-life WALL-E we've been waiting for. Retrieved from https://www.theverge.com/2016/6/27/12007772/anki-cozmo-robot-ai-toy-wall-e-pixar

Štuikys, V. (2015). Smart educational environments to teach topics in computer science: Theory, methodology and robot-based implementation. In *Smart Learning Objects for Smart Education in Computer Science: Theory, Methodology and Robot-Based Implementation* (pp. 1–315). Basel, Switzerland: Springer. https://doi.org/10.1007/978-3-319-16913-2

Sullins, J. P. (2008). Friends by design: A design philosophy for personal robotics

technology. In *Philosophy and Design* (pp. 143–157). Dordrecht, Netherlands: Springer. https://doi.org/https://doi.org/10.1007/978-1-4020-6591-0_11

Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, *26*(1), 3–20. https://doi.org/10.1007/s10798-015-9304-5

Sullivan, A., Kazakoff, E. R., & Bers, M. U. (2013). The wheels on the bot go round and round: Robotics curriculum in pre-kindergarten. *Journal of Information Technology Education: Innovations in Practice*, *12*(12), 203–219.

Sutton, K., Heathcote, A., & Bore, M. (2005). Implementing a web-based measurement of 3D understanding. In *Australasian Computer-Human Interaction Special Interest Group Conference*. Canberra, Australia. https://doi.org/10.1145/1108368.1108438

Sutton, K., & Williams, A. (2007a). Spatial cognition and its implications for design. In *International Association of Societies of Design Research*. Hong Kong, China. Retrieved from https://www.sd.polyu.edu.hk/iasdr/proceeding/papers/Spatial Cognition and its Implications for Design.pdf

Sutton, K., & Williams, A. (2007b). Spatial cognition and its implications for design. In *International Association of Societies of Design Research*. Hong Kong, China.

Swaid, S. I. (2015). Bringing computational thinking to STEM education. *Procedia Manufacturing*, *3*, 3657–3662.

https://doi.org/https://doi.org/10.1016/j.promfg.2015.07.761

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*, 257–285.

Tai, R., Liu, C. Q., Maltese, A. V., & Fan, X. (2006). Career choice: Enhanced: Planning early for careers in science. *Science*, *312*, 1143–1144.

Tao, J., & Tan, T. (2005). Affective computing: A review. In *Affective Computing and Intelligent Interaction* (Vol. Berlin; He, pp. 981–995). https://doi.org/10.1007/11573548_125

Tech, I. (2018). 2018 Summer tech courses. Retrieved from https://www.idtech.com/courses

Touretzky, D. S. (2017). Computational thinking and mental models: From kodu to calypso. *2017 IEEE Blocks and Beyond Workshop (B&B)*, 71–78. https://doi.org/10.1109/BLOCKS.2017.8120416

Triberti, S., Chirico, A., Rocca, G. La, & Riva, G. (2017). Developing emotional design: Emotions as cognitive processes and their role in the design of interactive technologies. *Frontiers in Pyschology*, *8*(October), 1–5. https://doi.org/10.3389/fpsyg.2017.01773

Um, E., Plass, J. L., Hayward, E. O., & Homer, B. D. (2012). Emotional design in multimedia learning. *Journal of Educational Psychology*, *104*(2), 485–498. https://doi.org/10.1037/ a0026609

Uttal, D. H., & Cohen, C. A. (2012). Spatial thinking and STEM education. When, why, and how? *Psychology of Learning and Motivation - Advances in Research and Theory, 57*, 147-181. https://doi.org/10.1016/B978-0-12-394293-7.00004-2

Uttal, D. H., Meadow, N. G., Tipton, E., Hand, L. L., Alden, A. R., Warren, C., & Newcombe, N. S. (2013). The malleability of spatial skills: A meta-analysis of training studies. *Psychological Bulletin*, *139*(2), 352–402. https://doi.org/10.1037/a0028446

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, Greenfoot, and Scratch - A discussion. *ACM Transactions on Computing Education*, *10*(4), 17.

Uysal, M. P., & Yalin, H. I. (2012). The effects of instructional software designed in accordance with instructional transaction theory on achievements of students. *International Journal of Human Sciences*, *9*(1), 185–204.

Vandenburg, S. G., & Kuse, A. R. (1978). Mental rotations, a group test of three-dimensional spatial visualization. *Perceptual and Motor Skills*, *47*(2), 599–604.

Vandevelde, C., Wyffels, F., Ciocci, M. C., Vanderborght, B., & Saldien, J. (2016). Design and evaluation of a DIY construction system for educational robot kits. *International Journal of Technology and Design Education*, *26*(4), 521–540. https://doi.org/10.1007/s10798-015-9324-1

Verner, I. M. (2004). Robot manipulations: A synergy of visualization, computation and action for spatial instruction. *International Journal of Computers for Mathematical*

Learning, *9*, 213–234.

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. https://doi.org/10.1007/s10639-015-9412-6

Wai, J., Lubinski, D., & Benbow, C. P. (2009). Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology*, *101*(4), 817–835. https://doi.org/10.1037/a0016127

Wainer, J., Feil-Seifer, David, J., Shell, D. A., & Mataric, M. J. (2006). The role of physical embodiment in human-robot interaction. *2006 IEEE ROMAN Proceedings of IEE International Symposium of Robot Human Interactive Communication*, *15th*(Hatfield, UK), 117–122. https://doi.org/10.1109/ROMAN.2006.314404

Weese, J.L., Feldhausen, R., & Bean, N. H. (2016). The impact of STEM experiences on student self-efficacy in computational thinking. *ASEE Annual Conference and Exposition, Conference Proceedings*, *2016-June*.

Weese, Joshua Levi. (2016). Mixed methods for the assessment and incorporation of computational thinking in K-12 and higher education. *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16*, 279–280. https://doi.org/10.1145/2960310.2960347

Wei, C.-W., Hung, I.-C., Lee, L., & Chen, N.-S. (2011). A joyful classroom learning system with robot learning companion for children to learn mathematics multiplication. *Turkish Online Journal of Educational Technology*, *10*(2), 11–23. Retrieved from https://www.learntechlib.org/p/53373/

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. https://doi.org/10.1007/s10956-015-9581-5

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. In *Proceedings of the First International Workshop on Computing Education Research* (pp. 13–24). Seattle, WA.

Williams, D. C., Ma, Y., Prejean, L., Ford, M. J., & Lai, G. (2007). Acquisition of physics content knowledge and scientific inquiry skills in a robotics summer camp. *Journal of Research on Technology in Education*, *40*(2), 201–216. https://doi.org/10.1080/15391523.2007.10782505

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on empty: The failure to teach K-12 computer science in the digital age. *Association for Computing Machinery*, 76. Retrieved from http://www.acm.org/Runningonempty/

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

Wing, J. M. (2008). Computational thinking and thinking about computing.

Philosophical Transactions of the Royal Society, 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2010). Computational thinking: What and why? Retrieved from https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education*, *18*(1), 1–20. https://doi.org/10.1145/3104982

Witherspoon, E. B., Schunn, C. D., Higashi, R. M., & Shoop, R. (2018). Attending to structural programming features predicts differences in learning and motivation. *Journal of Computer Assisted Learning*, *34*(2), 115–128. https://doi.org/10.1111/jcal.12219

Woods, S., Dautenhahn, K., & Schulz, J. (2004). The design space of robots: Investigating children's views. *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, 47–52. https://doi.org/10.1109/ROMAN.2004.1374728

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

Zeldin, A. L., & Pajares, F. (2000). Against the odds: Self-efficacy beliefs of women in mathematical, scientific, and technological careers. *American Educational Research*

Journal, *37*(1), 215–246. https://doi.org/http://doi.org/10.3102/00028312037001215