11-2018

Developing Computational Thinking with Educational Technologies for Young Learners

Yu-Hui Ching
Boise State University, yu-huiching@boisestate.edu

Yu-Chang Hsu
Boise State University, hsu@boisestate.edu

Sally Baldwin
Boise State University, SALLYBALDWIN@boisestate.edu

Publication Information

Ching, Yu-Hui; Hsu, Yu-Chang; and Baldwin, Sally. (2018). "Developing Computational Thinking with Educational Technologies for Young Learners". *TechTrends, 62*(6), 563–573. https://doi.org/10.1007/s11528-018-0292-7

Developing Computational Thinking with Educational Technologies for Young Learners

Yu-Hui Ching
Boise State University

Yu-Chang Hsu
Boise State University

Sally Baldwin
Boise State University

Abstract

This article aims to provide an overview of the opportunities for developing computational thinking in young learners. It includes a review of empirical studies on the educational technologies used to develop computational thinking in young learners, and analyses and descriptions of a selection of commercially available technologies for developing computational thinking in young learners. The challenges and implications of using these technologies are also discussed.

Introduction

In the last ten years, new educational technologies have been designed and developed to engage young learners (defined as pre-kindergarten to elementary school age learners) in computing and computational thinking activities along with the maker education movement and computer science initiatives (Smith, 2016). Computational thinking refers to a set of thinking skills, processes and approaches to solving complex problems by drawing on concepts from computer science (Wing, 2006). Computational thinking involves elements such as defining problems, collecting, analyzing and representing data, identifying and evaluating possible solutions against criteria, and generalizing problem-solving processes to other problems (International Society for Technology in Education (ISTE) & Computer Science Teachers Association (CSTA), 2011). Computational thinking, while it originates in computer science, can apply to any number of problem-solving contexts. Recently, computational thinking has been argued to be "an analytical ability essential to every child" (Resnick et al., 2009, p. 62). Wing (2006) declared, "To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (p. 33). It has also been suggested that computational thinking is a critical aspect of raising academic achievement and blending academics with real life for K-12 students (Sykora, 2014).

However, computational thinking is still relatively new to K-12 educators and educational researchers. Most of the existing empirical studies on the development of computational thinking have been conducted in high school contexts (Israel, Person, Tapia, Wherfel, & Reese, 2015). Educators and researchers have just begun efforts to introduce and develop computational thinking in young learners, and few research studies have focused on computational thinking in the kindergarten and elementary school settings (Saez-Lopez, Roman-Gonzaez, & Vazquez-Cano, 2016). This article provides an overview of the opportunities for developing computational thinking in young learners, a review of empirical studies on the educational technologies used to develop computational thinking, and the analyses of selected readily available technologies for computational thinking development in this age group. This article aims to offer multiple avenues for developing computational thinking in young learners, beyond traditional computers and computing devices, including toys and board games. The challenges and implications of using these educational technologies for computational thinking development will also be discussed.

Opportunities for Developing Computational Thinking in Young Learners

The desire to teach computational thinking to young learners dates back to Papert (1980), who declared, "…learning to communicate with a computer may change the way other learning takes place" (p. 6). Today's children have unprecedented access to computers or computing devices. The percentage of U.S. children with access to computers in their homes rose to 85% in 2012 (Child Trends, 2015). The American Academy of Pediatrics notes that children "spend more time with digital media than any other single influence" (Shifrin, Brown, Hill, Jana, & Flinn, 2015, p. 6). A cross-sectional study of exposure and use of mobile media devices by urban, low-income, minority children (ages six months to four years) indicates 96.6% of the children (N = 350) used mobile devices, often for entertainment (Kabali et al., 2015). Yet researchers emphasize that children's use of computers should go beyond consuming technology. Working with computers, not simply consuming media, can positively impact children's intellectual growth (Horn, Crouser, & Bers, 2012).

Developing Computational Thinking Through Programming

Researchers suggest children's best learning experiences with computers happen when they are designing, creating and inventing using computers (Papert, 1980; Resnick, 2002). To this end, computer programming is suggested for children of all ages. In this paper, "programming" refers to "the craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them" (Massachusetts Department of Elementary and Secondary Education (MDESE), 2016, p. 40) A "program" refers to "a set of instructions that the computer executes to achieve a particular objective" (MDESE, 2016, p. 40). The term "coding" has a narrower focus on "the act of writing computer programs in a programming language" (MDESE, 2016, p. 37). We aim to use these terms true to the definitions but recognize that the term "programming" and "coding" are often used interchangeably in research literature and daily conversations. The relationship between computational thinking and programming is still under discussion (NRC, 2010). In this paper, we consider "computational thinking" as a broad problem-solving framework involving skills, processes, and approaches to solve problems, and "programming" as a key practice for supporting and cultivating the cognitive tasks involved in computational thinking (Grover & Pea, 2013).

Through programming, children are able to use their creativity to command computers to solve problems. Programming has been identified as an extension of writing, and learning to program provides children with the skills to "write" new types of things, such as interactive stories, games, animations, and simulations (Resnick et al., 2009). When learning to program, young learners are not ''just learning to code, they are coding to learn'' (Resnick, 2013, para. 5). Through programming, they are learning to solve problems in systematic ways and are empowered to express ideas and themselves using a variety of tools (Sullivan & Bers, 2015).

Building and programming computational artifacts enables children to engage in high-level cognitive processes involving problem-solving, divergent thinking and reflection (Fessakis, Gouli, & Mavroudi, 2013; Kafai & Burke, 2014; Resnick, 2006). Research shows that learning how to program a computer can increase children's achievement in science, mathematics, language skills, and creative thinking (Horn et al., 2012). Using computers, kindergartners were able to learn simple programming and mathematical concepts, develop social skills, and practice problem-solving strategies (Fessakis, et al., 2013). Including basic coding in primary curricula helps develop and exercise young learners' general and higher-order thinking skills such as problem decomposition, analysis, and evaluation, which are critical to problem-solving (Falloon, 2016).

Educational Technologies in Computational Thinking Research Studies

Traditionally, learning programming begins with learning a programming language (e.g., Java, Python, C). Current efforts in educational technologies have made programming and the development of computational thinking more accessible and less abstract for young learners. Programming environments with the graphical and puzzle-like interfaces reduce the amount of reading required to navigate the options, and the need to learn syntax in order to give commands. These environments also shorten the feedback loops by allowing for immediate execution of commands (Fessakis et al., 2013).

A review of literature shows that at the pre-kindergarten to elementary level, computational thinking and programming is fostered with the use of robots (Gordon, Ackermann, & Breazeal, 2015; Martinez, Gomez, & Benotti, 2015; Sullivan, Elkin, & Bers, 2015), visual block-based programming software (Horn et al., 2012), electronic blocks (Wyeth, 2008; Wyeth & Wyeth, 2001), storybooks with stickers (Horn, AlSulaiman, & Koh, 2013) and devices controlled by buttons (Perlman, 1974). Morgado, Cruz, and Kahn (2010) described how ToonTalk, an animated, action-based programming software, could be used to overcome the limitations of children at the preliterate level. Kazakoff, Sullivan, and Bers (2013) provided preschoolers wit graphical blocks on a computer screen and tangible blocks to program LEGO® WeDo robots. Martinez et al. (2015) used a robot with a visual programming environment in research at the preschool and the elementary level. These researchers found that all learners could learn the computational concepts of sequence, loops, parameters, and conditional statements. Learners in elementary school could combine these concepts and apply them to program robots to move. In Flannery and Ber's (2013) study, learners who were four to six years old used Creative Hybrid Environment for Robotics Programming (CHERP) to teach a robot to dance the hokey pokey. Robolab (drag-and-drop graphical programming environment) were used by early elementary students to program LEGO® Mindstorms (Bers, Ponte, Juelich, Viera, & Schenker, 2002). Second and third graders created and programmed ancient Greek playgrounds using the LEGO® WeDo robot kit in a social science unit in Elkin, Sullivan, and Bers' (2014) study. The results of these empirical studies show strong evidence of the possibilities and positive outcomes for developing computational thinking through programming in young learners using various educational technologies.

Analyzing Educational Technologies for Computational Thinking Development

Recently, a growing number of educational technology products are targeted specifically at developing computational thinking and programming skills in young learners. These tools are most likely to encompass an enacting agent (e.g., a robot) and a programming tool featuring drag-and drop graphical interface for commanding the enacting agent. The use of a graphical programming interface enables learners to focus on the computational concepts instead of the syntax of programming languages.

To help educators and instructional designers get started using these tools with young learners, we analyzed a selection of currently available tools and developed a framework for analysis. This framework addresses two dimensions: the design of the tools (including the enacting agents and the programming software) and the pedagogical affordances. The first dimension examines whether the enacting agents and the programming tools are screen-based or tangibles. Kaifai and Burke (2014) pointed out that the educational technologies that engage children in programming have moved "beyond the computer screen to meld the digital with the tangible" (p. 91). The tangibles (i.e., the physical enacting agents and the manipulatives for programming) provide additional sensory input/output and immediacy. The manipulatives for programming enable learners to express their thinking and code through sorting and arranging manipulatives. This input approach makes programming accessible to young learners who have not yet learned how to use a keyboard, mouse or touch screen. The presence of a physical enacting agent allows learners to excitedly see the actual movement/consequence of the program occurring in the physical world. The immediate visual feedback of programming could help young learners more easily test their hypotheses and refine their ideas. Being able to touch and feel the enacting agents and the manipulatives for programming makes the programming activities more attractive for young learners who seek various sensory inputs.

Table 1 shows a selection of educational technologies classified by the existence of manipulatives for programming, and the existence of electronic physical enacting agents. In general, the tools that incorporate programming manipulatives are designed for the youngest learners (from pre-kindergarten to lower elementary grades).

Table 1. *Educational Technologies for CT Development.*

	Programming with manipulatives	No manipulatives for programming (Programming on screen)
Electronic Physical Enacting Agents	**Programming Toys** • Think & Learn Code-A-Pillar • Cubetto • Code & Go Robot Mouse Activity Set • KIBO	**Robot Kits** • Dash & Dot • LEGO® WeDo 2.0 • LEGO® Mindstorm
No Electronic Physical Enacting Agents	**Board Games** • Robot Turtles • Code Monkey Island **Augmented Reality Tools** • Osmo Coding	**Programming Concept Practicing Applications/Websites** • Kodable • Lightbot • Lightbot Jr. • Cargo-Bot • Code.org **Animation/Game Development Tools** • Scratch • Scratch Jr. • Hopscotch • Tynker: Coding for Kids

The second dimension emphasizes on the pedagogical affordances in the development of computational thinking. Brennan and Resnick (2012) devised a framework that has been adopted frequently to understand and assess computational thinking in education. The framework defines three critical dimensions of computational thinking: "computational concepts (the concepts designers employ as they program), computational practices (the practices designers develop as they program), and computational perspectives (the perspectives designers form about the world around them and about themselves)" (Brennan & Resnick, 2012, p. 3). Among these three computational thinking dimensions, we recognized that educational tools were designed to cultivate certain computational concepts, but the development of computational practices and computational perspectives lies in how educators and learners use the tools. As such, this paper focuses solely on analyzing the computational concepts that the selected tools are designed to cultivate in young learners. Table 2 further elaborates on the various computational concepts (Brennan & Resnick, 2012).

Table 2. *Computational Concepts.*

Concept	Description
Sequence	Creating a series of individual steps or instructions that can be executed by the computer
Loops	Running the same sequence multiple times
Parallelism	Making things happen at the same time
Events	One thing causing another thing to happen
Conditionals	Making decisions based on conditions
Operators	Supporting mathematical and logical expressions
Data	Storing, retrieving, and updating values

In the next section, we present an analysis of the selected tools followed by a summary table.

Programming Toys

The tools in this category typically target the youngest learners. To make programming activities easier for very young learners, these tools usually allow learners to directly control the toys without programming on screen using a computer or a computing device. These programming toys themselves are electronic physical agents that display the outcomes of programming. In addition, physical programming manipulatives are available to help younger learners organize and express their ideas. For example, children ages three to six can separate and re-connect the parts of Fisher-Price's Think & Learn Code-A-Pillar (http://www.fisher-price.com/en_US/brands/think-and-learn/index.html) to 'program' the toy to move in different directions or make lights and sounds. Code & Go Robot Mouse Activity Set (https://www.learningresources.com/product/learning+essentials--8482-+stem+robot+mouse+coding+activity+set.do) encourages children ages five and older to build a maze and then create a path for a programmable mouse to navigate. To program the mouse to move around, learners simply push the "forward," "backward," "left turn" and "right turn" buttons on the mouse. In addition, coding cards, serving as physical programming manipulatives, help learners organize their thoughts and chart the course before pushing buttons to enter a sequence of commands. Cubetto (https://www.primotoys.com/), designed for children ages three and up, engages very young learners in programming with a wooden toy-like robot, physical coding blocks and a control board. Learners arrange the coding blocks on the control board to command the robot to move around. KIBO (http://kinderlabrobotics.com/kibo/) is another tool designed for young learners ages four to seven years old, based on Bers and her colleagues' research on new technologies for young children (e.g., Flannery & Bers, 2013; Kazakoff, et al., 2013). With KIBO, learners have a set of wheels, motors and sensors to assemble onto the main robot body. Learners code by arranging the wooden blocks and then use the robot (with a built-in camera) to scan the barcodes on the programming blocks in order to execute the codes.

The aforementioned programming toys are designed for very young learners, from pre-school to lower elementary school levels. These toys mostly focus on teaching young children the computational concepts of sequence and loops. Among them, KIBO's programming capacity is more advanced, allowing the development of conditional statements. While Cubetto and KIBO are often considered as robot kits, we classify them as programming toys as they do not come with programming software for on-screen coding, like other robot kits reviewed in the next section do.

Robot Kits

The Toy Industry Association (2016) lists robots as one of the hottest trends in toys and play. Robots are appealing to children and to their parents, who view robots as educational toys with the ability to teach children "important concepts including coding, engineering, problem-solving and building" (Toy Industry Association, 2016, para. 5). Serving as manipulatives, robot kits allow young learners to "participate in creative explorations, develop fine motor skills and hand-eye coordination, and engage in collaboration and teamwork" (Elkin, Sullivan, & Bers, 2014, p. 155). Robot kits also present an engaging way to foster interdisciplinary explorations in young learners (Bers, 2008), involving the learning of science, technology, engineering, and mathematics (STEM) concepts, programming, computational thinking, and engineering skills (Benitti, 2012). In addition, research has shown that robots coupled with computer programming curriculum fostered computational thinking in young children (Sullivan & Bers, 2016). Robotics activities can situate learning of abstract computational concepts and problem-solving techniques and processes into more concrete experiences where students can create, observe and interact with physical objects for experiential learning (Petre & Price, 2004). With the help of age-appropriate robotics tools, "children as young as 4 to 6 years old can build and program simple robotics projects…. while also building their computational thinking skills" (Bers, Flannery, Kazakoff, & Sullivan, 2014, p. 145).

We reviewed three robot kits in this category. A robot and enabling programming software are included in each of the reviewed robot kits. Learners can use the programming software (usually block-based) on the computing devices to command the behavior of the robot (i.e., the electronic physical agent). For example, Wonder Workshops robots, Dash and Dot (https://www.makewonder.com) are pre-built robots that allow young children (five years and up) to program while they play. Learners use age-appropriate programming applications on a smartphone or tablet to program Dash and Dot's activities. LEGO® Mindstorms EV3 (https://www.lego.com/en-us/mindstorms) represents a widely-used robot kit for upper elementary to high school students. This robot kit allows designing and assembling of robots using LEGO® bricks and programming the assembled robots with a block-based programming software. The LEGO® WeDo

2.0 set (https://education.lego.com/en-us/shop/wedo%202) is aimed towards younger users (for children ages seven and up). Learners build robots with larger LEGO® bricks and less sensor options, and use a simplified block-based programming environment for coding. It has a curriculum packet specifically designed to develop learners' computational thinking skills (https://goo.gl/1s1j8j). For both LEGO® Mindstorms EV3 and LEGO® WeDo 2.0, STEM-related lesson plans are available on the LEGO® Education website (https://education.lego.com/en-us).

For these robot kits, the companion programming software is often powerful, allowing learners to develop multiple computational concepts, such as sequence, loops, events and conditionals. LEGO® robot kits involve learners in both physical (building with the LEGO® bricks) and digital building (programming) (Nash, 2017), presenting unique challenges for learners when they test their creations and troubleshoot. Additional support and guidance may be needed to help learners navigate both the physical and digital creation processes.

Board Games

Several board games are specifically designed to develop players' computational thinking skills. These board games do not include any electronic physical agents to enact the codes or commands. Programming manipulatives are available to express codes, usually in the format of cards. Players from three to eight years old can learn computational concepts when playing the board game Robot Turtles Game (http://www.robotturtles.com). Players use direction cards (manipulatives) to command turtle cards' movements toward a goal. As recommended by the game, an adult will manually move the turtle cards according to the players' directions. To play the game, players use the computational concepts of sequence and conditionals during the problem-solving process. Another board game, Code Monkey Island (http://codemonkeyplanet.com), was created for players ages eight years old and up. Players move the monkey figures around the board to a destination by applying computational concepts. This board game aims to help players learn computational concepts like conditional statements, loops, Boolean operators and logical operators.

Overall, board games that do not require the use of any computing devices are cost effective ways to develop computational thinking and even bring computational thinking into informal learning spaces such as homes, after-school programs or camps. Board games involve multiple people in gameplay, which is a great way to engage players in the discussion of computational concepts. One limitation of board games is that they are typically less comprehensive and only focus on a confined set of computational concepts.

Augmented Reality Tools

Augmented reality tools have been developed to make the coding experience more engaging and easier for younger learners. Osmo Coding Awbie Game (https://www.playosmo.com/en/coding/) teaches logic skills and problem solving with various game system kits played on iPads. Players can snap physical blocks together to create sequences of commands that can guide the character's movement. The Osmo system then scans the physical blocks and transforms them into codes to move characters toward the goal (reaching strawberries) on screen. The computational concepts sequence and loops are used during the problem-solving process.

Programming Concept Development Applications/Websites

A variety of programming applications have been designed to help learners develop programming concepts on mobile devices or on computers. Kodable (https://www.kodable.com) is targeted at children in kindergarten to fifth grade, and helps teach young learners programming concepts while working through 45 maze levels. Kodable also comes with lesson plans that are aligned with Computer Science standards for teachers or parents to help guide their learners in programming. Similarly, Lightbot Jr. (for learners ages four to eight) and Lightbot (for ages nine and up) (https://lightbot.com/flash.html) are programming applications that teach young learners how to plan, test, debug and code procedures and loops. Cargo-Bot, aimed at learners ages ten and up, teaches computational concepts such as sequence and loops. In an empirical study, Falloon (2015) found that Cargo-Bot might be efficient in developing a more technical understanding of computational concepts and might enable users to practice debugging during the process.

On Code.org, there is a rich collection of cloud-based programming concept development applications that help children start coding. Most of these applications start by asking learners to move an actor or character around by sequencing the code of movement with block-based programming software. As skills increase, learners can design their own animation to tell stories. For example, fans of the Disney movie *Frozen* may enjoy helping Elsa create patterns as she ice-skates, using the visual programming environment, Blockly, as part of the Code Studio (https://studio.code.org/s/frozen/stage/1/puzzle/1). Kalelioglu (2015) investigated 32 fourth graders learning programming using Code.org and found that students developed a positive attitude toward programming and both female and male students were successful in completing programming tasks. Kalelioglu concluded that the "Code.org site has motivating features, helpful instructional materials and lesson plans inside" (p. 209).

Generally, these programming concept development applications/websites usually ask learners to move an object around by sequencing the code of movements such as "forward," "backward," "left turn" and "right turn." The programming concepts of sequence and loops are addressed most frequently in these applications. These programming concept development applications/websites are easily accessible, but some tend to be limited in the numbers of computational concepts they aim to teach.

Animation/Game Development Tools

Animation/game development programming tools are also available for young learners to create computational artifacts. These tools allow learners to write codes to create animation or games and unleash their creativity and self-expression. Creating computational artifacts (e.g., animation, digital stories, and games) through programming anchors the learning of programming concepts and problem-solving processes in personally-meaningful contexts and increases the authenticity and relevancy to learners (Burke & Kafai, 2013). A recent literature review conducted by Kafai and Burke (2015) on 55 empirical studies of game-making revealed that a substantial number of studies reported that making games fostered the learning of programming and the development of computational concepts.

Scratch, a cloud-based and block-based programming environment developed by MIT (https://scratch.mit.edu/), is widely used by learners ages eight and up. Young learners have created a wide range of games and interactive projects using this tool (Burke & Kafai, 2013). Kafai and Peppler (2011) randomly sampled 534 user projects from the Scratch website and found a broad diversity in the design of the computational artifacts, with half of them focusing on interactive narrative projects, such as animations, interactive art, and narrative games. Kafai and Peppler's study suggests that there are multiple pathways into using Scratch (Burke & Kafai, 2013), and engaging in programming and problem-solving processes. In addition, Brennan and Resnick (2012) developed their computational thinking framework based on learners' work in the Scratch environment.

ScratchJr, (http://www.scratchjr.org), a spin-off of Scratch, is a visual programming environment created for children ages five to seven to use on tablets. With ScratchJr, younger learners drag and drop blocks to create digital stories. Research has found that young learners (five to six years old) used a range of general computational and higher-order thinking skills when engaging in coding tasks using ScratchJr (e.g., Falloon, 2016).

Some programming applications capitalize on learners' motivation in playing and creating digital games. A literature review identified that game-making (Hayes & Games, 2008) enables, among others, four major different learning goals including learning programming, making programming more engaging for females and underrepresented learners, learning content in other academic domains, and understanding design concepts. Hopscotch (http://www.gethopscotch.com) is a free application for the iPad that helps children develop their own games, stories and animations using visual blocks. Learners can also download and remix games made by other learners. Starter tutorials are available for new users. Tynker (https://www.tynker.com) is a learning system that teaches young learners to code. Learners can use visual blocks to design games or program for robotics and drones.

These animation/game development tools are usually powerful in developing young learners' computational concepts. Depending on the level of complexity of the computational artifacts to be created, young learners have the opportunity to apply most of the computational concepts identified by Brennan and Resnick (2012) in their codes. Young children learn problem-solving skills by experimenting with different ways to program and iteratively revising their programs for creative expression.

The table below summarizes the tools we previously introduced. These are commercial tools that are readily available for free or for purchase. The affordance of these tools for computational concepts development is highlighted in the table and is determined based on the official description of the tools, the findings from empirical research and the authors' experiences using the tools. The classification is by no means comprehensive but serves as a starting point for educators interested in exploring and learning more about computational thinking tools. Creative use of these tools is likely to broaden a learner's computational thinking development in unique ways.

Table 3. *An Analysis of Educational Technologies for CT Development.*

Tool	**Manufacturer/ Price**	**Age Group**	**EPA**[1]	**MP**[2]	**Computa-tional Concept**	**Programming Activity Description**
Programming Toys (Including an electronic physical enacting agent and manipulatives for programming; Targeting at the youngest learners; Focusing mostly on sequence and loops)						
Think & Learn Code-A-Pillar	Fisher Price/ $35.89	2-4	Y	Y	Sequence	Learners separate and re-connect the pars of the caterpillar to 'program' the toy to move in different directions or make lights and sounds.
Cubetto	Primo Toys/ $225.00	3+	Y	Y	Sequence Loops	Learners build a maze and create a path for a programmable mouse to navigate. Learners program the mouse to move around, by pushing the direction buttons on the mouse. Coding cards (physical programming manipulatives) are available.
Code & Go Robot Mouse Activity Set	Learning Resources/ $59.99	5+ (5-9)	Y	Y	Sequence Loops	Learners program a wooden toy-like robot to move around by arranging physical coding blocks on a control board.
KIBO	KinderLab Robotics/ Starting from $229.00	4-7	Y	Y	Sequence Loops Conditionals	Learners assemble a robot with a set of wheels, motors and sensors, and code by arranging the wooden blocks. Learners use the robot (with a built-in camera) to scan the barcodes on the programming blocks in order to execute the codes.
Robot Kits (Including a robot and programming software; Coding on screen; Powerful programming software allowing the development of multiple computational concepts, such as sequence, loops, events and conditionals)						
Dash & Dot	Wonder Workshops/ $199.95	5+	Y	N	Sequence Loops Events Conditionals	Learners use block-based programming applications on a smartphone or tablet to program Dash and Dot's activities such as moving around, making sound, reacting to sensory input.
LEGO® Mindstorms EV3	LEGO®/ $349.99	10-15	Y	N	Sequence, Loops Events Conditionals Operators Data Parallelism	Learners design and assemble robots using LEGO® bricks and program the assembled robots with a block-based programming software.

LEGO® WeDo 2.0	LEGO®/ $179.95	7+	Y	N	Sequence Loops Events Conditionals	Younger learners build robots with larger LEGO® bricks, and use a simplified block-based interface for coding. The Product website has curriculum packet specifically designed to develop learners' computational thinking skills.
Board Games (Cost effective; No need for computing devices; No electronic physical agents to enact the codes or commands; Programming manipulatives in the format of cards; Limited in scope; Great for informal learning spaces with multiple players)						
Robot Turtles Game	Thinkfun/ $18.95	3-8	N	Y	Sequence Loops	Players use direction cards (manipulatives) to command turtle cards' movements toward a goal. An adult will manually move the turtle cards according to the learners' directions.
Code Monkey Island	Code Monkey Games/ $19.99	8+	N	N	Loops Conditionals Operators	Players move the monkey figures around the board to a destination while applying computational concepts.
Augmented Reality Tool (Programming with physical manipulatives)						
Osmo Coding Awbie Game	Osmo/ $49.99	5-12	N	Y	Sequence Loops	Players can snap physical blocks together to create sequences to guide the character. The Osmo system then scans the physical blocks and transforms them into codes to control characters on screen.
Programming Concept Practicing Applications/Websites (Programming on-screen; Used mostly to develop computational concepts of sequence and loops; Limited in scope)						
Kodable	Surfscore/ $29.00	5-11	N	N	Sequence Conditionals Loops Operators	Learners move objects through mazes by giving directions.
Lightbot	LightBot, Inc./ $4.99	9+	N	N	Sequence Loops	Learners give directions to move a robot around (on screen) to achieve certain goals.
Lightbot Jr.	LightBot, Inc./ Free	4-8	N	N	Sequence Loops	Learners give directions to move a robot around (on screen) to achieve certain goals.
Cargo-Bot	Two Lives Left/ Free	10 and up	N	N	Sequence Loops	Learners give directions to sort or move cargo boxes around to achieve certain goals.
Code.org	Code.org/ Free	5-18	N	N	Various but mostly sequence and loops	Learners move an actor or character around by sequencing the code of movement with block-based programming environments. Learners can also design their own animation to tell stories.

Animation/Game Development Tools (Programming on screen; Most powerful and versatile for computational concept development)						
Scratch	MIT media lab/ Free	8 and up	N	N	Sequence Loops Conditionals Events Operators	Learners program using the cloud-based and block-based programming environment to create animations, digital stories, or interactive games.
Scratch Jr.	DevTech Research Group (Tufts) & Lifelong Kindergarten Group (MIT) and Playful Invention Company/ Free	5-7	N	N	Sequence Loops Events Operators	This is an app available on tablets. Young leaners use their fingers to drag and drop blocks to create digital stories and games.
Hopscotch	Hopscotch Technologies/ Free with in- App Purchase	9-13	N	N	Sequence Loops, Conditionals Events Operators	Learners develop their own games, stories, and animations using visual blocks. Learners can also download and remix games made by other learners. Starter tutorials are available for new users.
Tynker: Coding for Kids	Tynker/ Free with in- App Purchase	9-11	N	N	Sequence Loops, Conditionals Events Operators	Learners can use visual blocks to design games or program for robots and drones.

[1]Electronic Physical Agent
[2]Manipulative for Programming

Discussion

Computational thinking is considered a fundamental skill, similar to literacy and mathematics, necessary to navigate an information society. The development of computational thinking in young learners may build a foundation for learners to embrace a systematic problem-solving approach and foster higher-order thinking skills (e.g., Fallon, 2016). Developmentally appropriate programming tools can help young children learn computational thinking and reasoning (Bers et al., 2014). Being exposed to STEM and computer programming at a young age may also help them overcome social and gender stereotypes when choosing STEM careers (Metz, 2007; Sanford & Madill, 2007) and alleviate obstacles to entering these academic fields (Madill et al., 2007; Markert, 1996).

Although an increasing number of educational technologies offer unprecedented opportunities for engaging young learners in computational activities and thinking, challenges exist. Educational technologies for computational thinking development can be costly for schools. Some technologies, such as robot kits, are more capable in terms of allowing for complicated programming activities and computational concept development. However, they also tend to have steep learning curves and require more time invested upfront by learners and educators. Other technologies such as board games and programming concept development applications, are easy to use, but limited in pedagogical affordances. In addition, learners can outgrow a singular method of instruction quickly.

In regards to the ability of young learners to learn computational thinking concepts, some aspects of computational thinking can be particularly challenging. For example, Bers et al. (2014) found that the concept of “control flow” in programming (e.g., loops and conditional statement) is more abstract and thus, fewer kindergarten students were able to master it. They suggested more time should be devoted to the more complicated concepts in the curriculum to allow children to fully understand them. Kalelioglu (2015) and Fessakis et al. (2013) identified that some students face difficulties when visual-spatial abilities were needed to determine the appropriate directions to move objects around through programming. Falloon (2016) found that debugging presents challenges to young learners. Learners may get frustrated during the process and adopt less systematic strategies if they are not initially successful.

Marrying effective pedagogies with technologies to introduce computational thinking to young learners is likely to enhance and inspire creativity and problem-solving instead of frustration and discouragement. Age-appropriate curriculum and technologies are of key importance. Falloon (2016) noted, "Well-designed coding tasks embedded in collaborative, problem-based and/or thematic curriculum designs, can support a wide array of student thinking capabilities" (p. 591). For example, Bers et al. (2014) paired robotic programming activities with songs (e.g., "Hokey Pokey") and games (e.g., "Simon Says") to engage kindergarten students. Lu and Fletcher (2009) suggested that computational thinking should be introduced to young learners by establishing important vocabulary and symbols that can be used to describe computation and abstraction. Fessakis et al. (2013) also pointed out that the critical point to the systematic integration of programming in kindergarten lies in "the development of appropriately designed learning activities and supporting material, which would have been applied and verified and could be easily integrated in everyday school practice by well informed and prepared teachers" (p. 89).

Like other technology in the school, computational thinking development tools often need to be shared in small groups, which could provide great opportunities for collaboration and communication. These opportunities also increase the need for learners to articulate and make their computational thinking (e.g., skills, processes and approaches used to solve problems) visible for peers and teachers. Meaningful teachable moments occur during the articulation process. Educators who are interested in bringing computational thinking into their classrooms can start with those tools that come with curriculum and community support (e.g., Scratch or Kodable). Computational thinking is still new to pre-K to sixth grade education. It is likely that teachers need substantial professional development in selecting appropriate technologies and learning how to program so that they feel confident in using the tools on their own and in the classroom.

Future Directions

While many new educational technologies for computational thinking development are readily available, future directions for research lie in designing and developing effective curriculum that can help integrate these tools to foster computational thinking development in young learners. In addition, combining computational thinking development with content disciplines is likely to help young learners see the real-world application of computational thinking. Most of the existing technologies and accompanying curriculum/lesson plans focus mainly on the development of programming skills. In the future, the curriculum could be designed to offer students the opportunity of using computational thinking to solve community problems or complete STEM-related projects. A recent study conducted by Israel et al. (2015) examined different models of integrating computational thinking into a variety of teaching contexts. While four teachers successfully embedded computing into their core curricula, the authors pointed out that further research is needed to help teachers successfully collaborate in designing curricula that integrate computational thinking within content areas for real-world applications.

Designers of educational technologies may focus on ideas that make the computational thinking development conducive for young learners. Bers et al. (2014) noted that many of the challenges arising in their study were posed by the robotics hardware itself and stressed the importance of making developmentally appropriate hardware and software specifically designed for young children. For example, CHERP programming language enables children to create programs to control robots from tangible wooden blocks and/or graphical on-screen icons. The availability of the hybrid interface offers younger learners the opportunity to select developmentally appropriate means for expressing their thoughts. Designers may also want to establish and embed a systematic problem-solving or trouble-shooting process (e.g., problem formulation, solution expression and evaluation) into the technologies to help scaffold the learning process for young learners (Repenning, Basawapatna, & Escherle, 2017).

Although research on using robot kits and animation/game development tools in developing young learners' computational thinking are available, more research is needed in examining the effective use of other educational technologies for developing computational thinking in young learners. Furthermore, we only discussed the pedagogical affordance of the selected technologies on the aspects of computational concepts. Future studies may explore how various tools can be best used to develop computational practices and perspectives as defined in Brennan and Resnick's (2012) framework.

Conclusion

Computational thinking has been promoted as an important and fundamental skill for learners of all ages. Educators and researchers have just begun to explore ways to develop this fundamental skill in young learners. This paper introduces and reviews educational technologies that have been designed and/or researched for teaching computational thinking skills to young learners. The pedagogical affordances of the tools in developing computational concepts (Brennan & Resnick, 2012) have also been discussed. It is our goal to provide a road map for educators to navigate this new but critical area for development in young learners. Given the educational goals, resources and ages of the learners, educators and instructional designers can use this article as a guide to support their exploration and selection of age-appropriate technologies for their learners.

References

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic Review. *Computers & Education, 58,* 978-988.

Bers, M. (2008). *Blocks to robots: Learning with technology in the early childhood classroom.* New York, NY: Teachers College Press.

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

Bers, M., Ponte, I., Juelich, K., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics in early childhood education [Electronic version]. *Information Technology in Childhood Education Annual, 2002*(1), 123-145.

Burke, Q., & Kafai, Y. B. (2013). A decade of game-making for learning: From tools to communities. In H. Agius & M. C. Angelides (Eds.), *The handbook on digital games* (pp. 689-709). Hoboken, NJ: Wiley-IEEE Press.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Paper presented at Annual American Educational Research Association meeting, Vancouver, BC, Canada.

Child Trends. (2015). Home computer access and Internet use. Retrieved from http://www.childtrends.org/indicators/home-computer-access/

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom [Electronic version]. *Journal of Information Technology Education: Innovations in Practice, 13,* 153-169. Retrieved September 27, 2017, from http://www.jite.org/documents/Vol13/JITEv13IIPvp153-169Elkin882.pdf

Falloon, G. W. (2015). What's the difference? Learning collaboratively using iPads in conventional classrooms. *Computers & Education, 84,* 62-77.

Falloon, G. W. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. on the iPad. *Journal of Computer Assisted Learning, 32,* 576-593.

Fessakis, 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

Flannery, L. P., & Bers, M. U. (2013). Let's dance the "robot hokey-pokey!" Children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education, 46*(1), 81-101.

Gordon, M., Ackermann, E., & Breazeal, C. (2015, March). Social robot toolkit: Tangible programming for young children. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts* (pp. 67-68). New York, NY: ACM.

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

Hayes, E. R., & Games, I. A. (2008). Making computer games and design thinking: A review of current software and strategies. *Games and Culture, 3,* 309–322.

Horn, M. S., AlSulaiman, S., & Koh, J. (2013, June). Translating Roberto to Omar: Computational literacy, stickerbooks, and cultural forms. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 120-127). New York, NY: ACM.

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: The case for a hybrid approach [Electronic version]. *Personal and Ubiquitous Computing, 16*(4), 379-389.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education, 82,* 263-279.

ISTE & CSTA (2011). Operational definition of computational thinking for K-12 education. Retrieved September 27, 2017, from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2

Kabali, H. K., Irigoyen, M. M., Nunez-Davis, R., Budacki, J. G., Mohanty, S. H., Leister, K. P., & Bonner, R. L. (2015). Exposure and use of mobile media devices by young children [Electronic version]. *Pediatrics, 136*(6), 1044-1050.

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.

Kafai, Y. B. & Burke, Q. (2015) Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313-334. doi: 10.1080/00461520.2015.1124022

Kafai, Y. B., & Peppler, K. A. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. *Review of Research in Education*, *35*(1), 89-119. doi: 10.3102/0091732X10383211

Kalelioglu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200-210.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, *41*(4), 245-255.

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin, 41*(1), 260-264.
Massachusetts Department of Elementary and Secondary Education (MDESE). (2016, June). *2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework*. Malden, MA: Author. Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf

Madill, H., Campbell, R. G., Cullen, D. M., Armour, M. A., Einsiedel, A. A., Ciccocioppo, A. L....Coffin, W. L. (2007). Developing career commitment in STEM-related fields: Myth versus reality. In R. J. Burke, M. C. Mattis, & E. Elgar (Eds.), *Women and minorities in science, technology, engineering and mathematics: Upping the numbers* (pp. 210-244). Northampton, MA: Edward Elgar Publishing.

Markert, L. R. (1996). Gender related to success in science and technology [Electronic version]. *The Journal of Technology Studies, 22*(2), 21-29.

Martinez, C., Gomez, M. J., & Benotti, L. (2015). A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 159-164). New York, NY: ACM.

Metz, S. S. (2007). Attracting the engineering of 2020 today. In R. Burke & M. Mattis (Eds.), *Women and minorities in science, technology, engineering and mathematics: Upping the numbers* (pp. 184-209). Northampton, MA: Edward Elgar Publishing.

Morgado, L., Cruz, M., & Kahn, K. (2010). Preschool cookbook of computer programming topics. *Australasian Journal of Educational Technology*, *26*(3), 309-326.

Nash, J. (2017, June 1). Coding in the classroom with real-world learning. Retrieved September 27, 2017, from https://www.iste.org/explore/articleDetail?articleid=980&category=Innovator-solutions&article=Coding+in+the+classroom+with+real-world+learning

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: The National Academies Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books, Inc.

Perlman, R. (1974). *TORTIS (Toddler's Own Recursive Turtle Interpreter System).* Cambridge, MA: Massachusetts Institute of Technology A.I. Laboratory. Retrieved September 27, 2017, from http://files.eric.ed.gov/fulltext/ED118366.pdf

Petre, M., & Price, B. (2004). Using robotics to motivate “back door” learning [Electronic version]. *Education and Information Technologies*, *9*(2), 147-158.

Repenning, A., Basawapatna, A. R., & Escherle, N. A. (2017). Principles of computational thinking tools. In P. J. Rich and C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking, educational communications, and technology: Issues and innovations* (pp. 291-305). Cham, Switzerland: Springer.

Resnick, M. (2002). Rethinking learning in the digital age. In G. Kirkman (Ed.), *The global information technology report: Readiness for the networked world* (pp. 32-37). Oxford, England: Oxford University Press.

Resnick, M. (2006). Computer as paintbrush: Technology, play, and the creative society. In D. Singer, R. Golikoff, and K. Hirsh-Pasek (Eds.), *Play=learning: How play motivates and enhances children's cognitive and social-emotional growth* (pp.192-208). Oxford, England: Oxford University Press.

Resnick, M. (2013, May 8). Learn to code, code to learn. *EdSurge*. Retrieved September 27, 2017, from https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn

Resnick, M., Maloney, J., Monroy-Hernandez, 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*(11), 60-67. Retrieved September 27, 2017, from http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf

Saez-Lopez, J., Roman-Gonzaez, M., & Vazquez-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.

Sanford, K., & Madill, L. (2007). Understanding the power of new literacies through video game play and design. *Canadian Journal of Education/Revue Canadienne de l'éducation*, *30*(2), 432-455.

Shifrin, D., Brown, A., Hill, D., Jana, L., & Flinn, S. K. (2015). Growing up digital: Media research symposium. *American Academy of Pediatrics*, *1*, 1-7.

Smith, M. (2016). *Computer science for all*. Washington, DC: Office of Science and Technology Policy, Executive Office of the President. Retrieved from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

Steele, C. M. (1997). A threat in the air: How stereotypes shape intellectual identity and performance [Electronic version]. *American Psychologist*, *52*(6), 613-629.

Sykora, C. (2014, September 11). Computational thinking for all. Retrieved September 27, 2017, from https://www.iste.org/explore/articleDetail?articleid=152

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 [Electronic version]. *International Journal of Technology and Design Education*, *26*(1), 3-20.

Sullivan, A., Elkin, M., & Bers, M. U. (2015, June). KIBO robot demo: Engaging young children in programming and engineering. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 418-421). New York, NY: ACM.

Toy Industry Association, Inc. (2016, February 14). Top toy trends of 2016 announced by Toy Industry Association (TIA), the official voice of the Toy Fair. Retrieved September 27, 2017, from http://www.toyassociation.org/PressRoom2/News/2016_News/Top_Toy_Trends_of_2016_Announced_by_Toy_Industry_Association__TIA____the_Official_Voice_of_Toy_Fair.aspx#.WNPzoxiZNsN

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks [Electronic version]. *The Journal of the Learning Sciences*, *17*(4), 517-550.

Wyeth, P., & Wyeth, G. F. (2001). Electronic blocks: Tangible programming elements for preschoolers. In M. Hilrose (Ed.), *IFIP TC13 International Conference on Human-Computer Interaction* (pp. 496-503). Amsterdam, The Netherlands: IOC Press.