A Survey of Computational Kits for Young Children

Junnan Yu
University of Colorado Boulder
Boulder, CO USA
junnan.yu@colorado.edu

Ricarose Roque
University of Colorado Boulder
Boulder, CO USA
ricarose@colorado.edu

ABSTRACT
This paper presents a survey of computational kits that enable young children (ages 7 years old and under) to explore computing ideas and practices. We examined physical, virtual, and hybrid kits across three different perspectives: how they are designed, how they support children to explore computational concepts and practices, and how they enable children to engage in a range of projects and activities. Based on our analysis, we present design suggestions and opportunities to expand the possibilities in how children can engage in computing, what kinds of projects children can make, and what kinds of computational ideas children can explore.

Author Keywords
Coding; early childhood; computational thinking.

ACM Classification Keywords
• Human-centered computing ~ User interface toolkits • Applied computing ~ Interactive learning environments

INTRODUCTION
For more than a decade, education researchers, policymakers, and industry leaders have recognized the importance of helping young people cultivate computational thinking, or the ability to use concepts from computer science to solve problems and understand the world in new ways [37,54]. These concepts include how to think algorithmically, to break down complex ideas into smaller parts, or to uncover issues or "bugs" in instructions. Jeannette Wing who popularized the phrase argues that computational thinking "represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" [54]. As more of our everyday activities are influenced by computing, such as transportation, banking, and entertainment, computational thinking can be useful for everyone to participate in our digitally mediated society.

While many technologies have emerged to support youth in exploring computational concepts and practices, opportunities in early childhood are especially promising to cultivate interests early in computing as well as to support the development of social, emotional, and cognitive milestones [3,9]. Many studies have shown that early interventions can have compounding effects later in life and influence personal and academic outcomes [33]. Early exposure can also help mitigate barriers to participation in computing. For example, Master et al. [32] found that girls as young as 6 years old believe that boys are better at robotics and programming, but exposure to coding and robotics can moderate these stereotypes and help to improve their sense of self-efficacy.

In this paper, we present a survey of computational kits and toys, which we define as technologies that were designed to enable children to explore computational concepts and practices. We collected kits that were developed in academic and/or commercial contexts and examined them across three perspectives: their design features, their support for the exploration of computational concepts and practices, and their expressivity, or how well a kit supports various projects and explorations. We end by discussing the opportunities for designing computational kits for young children to cultivate computational thinking.

METHODS

Collecting kits
We examined computational kits from both academic and commercial venues in this survey. The earliest designs of computational kits for children date back to the 1970s and started in research labs. The Logo turtle robot and Slot Machines are two of the earliest kits that supported children to explore computational thinking [35]. In the recent decade, many research projects of computational kits have gone out of research labs and become publicly accessible through museum exhibits or commercial products. Because of this history in academia, we began our review by identifying some representative research projects of computational kits for young children, such as ScratchJr [16], Tern [22], and KIBO [50]. Then we examined references from their published papers to discover more kits. To broaden our survey, we searched for publications in the ACM digital library using keywords from the papers of these representative projects, such as "computational toys", "programming", "early childhood", "education", and "STEM". To find existing kits in commercial venues, we searched for kits on online product platforms Amazon and Target. We used the following keywords, as recommended by Amazon, to search for kits: "coding toys", "programming

IDC '18, June 19–22, 2018, Trondheim, Norway

ACM ISBN 978-1-4503-5152-2/18/06…$15.00
https://doi.org/10.1145/3202185.3202738

toys", "coding for kids", and "STEM toys". We found kits on Target using pre-defined categories on their platform, such as "coding and STEM" [51]. Based on reviewer feedback, we also looked into kits in Purdue 2017 Engineering Gift Guide [26] and Ehsan and colleagues' survey of computational apps [13]. The Purdue 2017 Engineering Gift Guide is a collection of toys, games, books, and apps designed for promoting engineering thinking and design in children ages 3-18. We reviewed kits that were labeled with "Coding/Programming" and "Computational Thinking". Some of the surveyed kits in literature have already been commercialized or made public, such as KIBO, roBlocks [47], and ScratchJr. For such kits, our analysis takes into consideration both the published papers about these kits and the information provided by sellers or the project's product page.

Filtering kits

Among the collected kits, we curated a set of kits that met the following two criteria: (1) the recommended age group was 7 years old and under; and that (2) one of the claimed goals of a kit was to enable children to explore computer programming, computational thinking, or other computer science concepts and practices. To focus our analysis, we also excluded the following: kits that do not provide a recommended age group, such as Blockly Games [18]; and kits that may involve certain computational thinking but the main purpose is not centered on computational skills, such as Topobo [44], a construction kit with kinetic memory, and System Blocks [57] which is focused on system dynamics simulation.

Based on the two filtering criteria, we selected 34 kits, which are described in Table 1. We organized the kits based on their physical features and categorized them as: physical, virtual, and hybrid kits. Physical kits are the kits whose components are all tangible, such as KIBO, which consists of a physical robot and a set of tangible programming blocks. Physical kits can be further divided into physical kits with electronics and physical kits without electronics, such as the board game Robot Turtles [48]. Virtual kits are PC and/or mobile-device-based applications without physical components, such as ScratchJr, a tablet-based application for young children to create interactive stories. Hybrid kits consist of both physical and virtual parts, such as Strawbies [24], which asks children to play a video game by manipulating tangible command tiles. Hybrid kits can also be classified into two subcategories based on the form of programming blocks: kits that consist of virtual programming blocks, or kits that consist of tangible programming blocks.

Physical kits with electronics	Curlybot [17], Tangible Programming Bricks [34] Electronic Blocks [56], roBlocks [47] or Cubelets [36], Dr. Wagon [6], KIBO [50], Torino [52], Cubetto [42], Plobot [41], Robot Mouse [29], Code-a-pillar [15], Bee-Bot [53], Pro-Bot [58]
Physical kits without electronics	Robot Turtles [48], Hello Ruby [31]
Virtual kits	ScratchJr [16], LightBot [30], Move the Turtle [38], Bitsbox [59], Cargo-Bot [60], Codeable Crafts [61], RoboZZle [62], Run Marco! [63], The Foos [64]
Hybrid kits with virtual programming blocks	Dash & Dot [55], Cozmo + Code Lab [65], Thymio Robot [66], meeperBOTS [67], COJI [68]
Hybrid kits with tangible programming blocks	Digital Dream Lab tabletop puzzle block system [40], Strawbies [24], Roberto [20], Puzzlets [12], Blue-Bot [69]

Table 1. The analyzed computational kits in this paper

Analyzing kits

We examined the remaining kits from three perspectives: design features, computational thinking, and expressivity.

In the first perspective, we focused on a kit's design features. Physical features can strongly influence how children perceive and use a computational kit. For example, in a study comparing tangible and virtual programming interfaces, Horn et al. [21] found that children were more likely to engage with tangible interfaces. In another study comparing tangible and virtual programming environments, Horn and colleagues [19] suggested that hybrid interfaces can enable children and educators to decide for themselves what makes the most sense for their particular situation.

In the second perspective, we examined how a kit supported computational concepts and practices using the computational thinking framework by Brennan and Resnick [5]. This framework included seven computational concepts (Sequences, Events, Parallelism, Loops, Conditionals, Operators, and Data) and four computational practices (being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing), which are briefly summarized in Table 2 and Table 3, respectively. While there have been many discussions and multiple definitions of computational thinking [27], Brennan and Resnick's framework is based on and includes concrete experiences of children creating with programming, particularly in the Scratch environment. For instance, Brennan and Resnick used an example of programming a cat sprite in Scratch to move to illustrate Sequences in the framework. By putting together a sequence of motion programming blocks, the cat can move across the screen based on the coding instructions

[5]. These concrete examples of children's programming experiences helped to examine how kits may support different computational thinking concepts and practices.

Concept	Description
Sequence	Identifying a series of steps for a task
Loops	Running the same sequence multiple times
Events	One thing causing another thing to happen
Parallelism	Making things happen at the same time
Conditionals	Making decisions based on conditions
Operators	Identifying a series of steps for a task
Data	Storing, retrieving, and updating values

Table 2. Computational concepts

Practices	Description
Experimenting and iterating	Developing a little bit, then trying it out, then developing more
Testing and debugging	Making sure things work – and finding and solving problems when they arise
Reusing and remixing	Making something by building on existing projects or ideas
Abstracting and modularizing	Exploring connections between the whole and the parts

Table 3. Computational practices

In the third perspective, we examined how expressive a kit can be. We define expressivity as how well a kit enables kids to create a wide range of projects. In their reflection on designing construction kits for kids, Resnick and Silverman highlighted the importance of kits having "wide walls", or enabling children to explore and create across a range of possibilities to express their many ideas and interests [45]. To evaluate the expressivity of a kit, we examined the activities the kit supports and the openness of these activities—whether these activities are predefined by the designers (e.g. kids must follow a predetermined challenge or set of instructions) or if kids can define their own goals and projects (e.g. kids can explore different possibilities with a kit).

For each perspective, we analyzed the information provided in published papers, a kit's project website, and/or a product page from the seller.

Limitations

The potential of a kit to support computational thinking or a range of activities largely depends on how children and facilitators use it. Our analysis of a kit is based on the description by its designers, which may not necessarily reflect what might happen when children interact with it. A playtest will be needed to evaluate a kit's intentions. Our analysis, therefore, may overestimate or underestimate a kits' potential as a tool to support computational thinking or to enable the exploration of a wide range of projects. In addition, even though we tried to review as many kits as possible in this study, we understand that we may have missed some computational kits which satisfy our filtering criteria. However, we believe our curated set of kits is sufficient to help us explore commonalities as well as imagine new possibilities.

In the next section, we use these three perspectives to identify common design strategies and features across the kits. In the Discussion, we build on this survey to highlight opportunities for further exploration of computational kits for young children.

FINDINGS

In this section, we examine the surveyed kits through three perspectives. The first perspective explores the design features of the kits. The second perspective examines what kinds of computational concepts and practices are supported by different kits, and how these concepts and practices are supported. The third perspective focuses on the expressivity of the kits.

Perspective 1: Examining design features

Typically, a computational kit consists of a set of programming blocks, a robot or sprite controlled by the programming blocks, and some supporting materials. For perspective 1, we analyzed the characteristics of the programming blocks, programmed robots or sprites, and supporting materials.

Programming syntax and semantics

All the surveyed kits either use physical blocks or graphical blocks to create programs. For most physical kits and hybrid kits, the programming blocks and robot are separated, but there are a few kits with programming blocks embedded in the robot in the form of buttons. The Bee-Bot [50] and Blue-Bot [69], for example, have directional buttons on the top of the bee robots and children can press these buttons to program its movement, see in Figure 1 (a). The concept of a block as a "puzzle piece" is a frequently-used form for both tangible and graphical programming blocks, such as the programming blocks of Strawbies, ScratchJr, Cozmo+Code Lab [65], Codeable Crafts [61], and Run Marco! [63]. Figure 1 (b) and (c) show the puzzle-form programming blocks of Strawbies and ScratchJr, respectively.

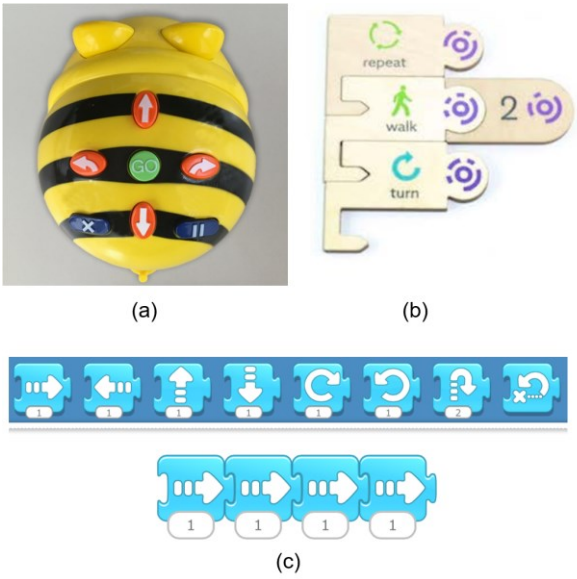

Figure 1. The programming blocks of (a) Bee-Bot [53]**, (b) Strawbies** [24]**, and (c) ScratchJr** [70]**.**

Most of the programming commands included by kits involve motion such as moving forward or turning left. Some kits include other command blocks such as function blocks, data blocks, and special-effect blocks. For example, in Cubetto [42], children can create a small code block and call the block in the main function through a function block, as shown in Figure 2. roBlocks includes sensor blocks of light, sound, touch, and distance which children can use to build an interactive robot. Strawbies allows children to create special animations for the game sprites, such as tornado and flashlight effects. In addition to basic motion commands, COJI [68] also allows children to program the expression of the COJI robot through inserting (a sequence of) emojis in a block of code.

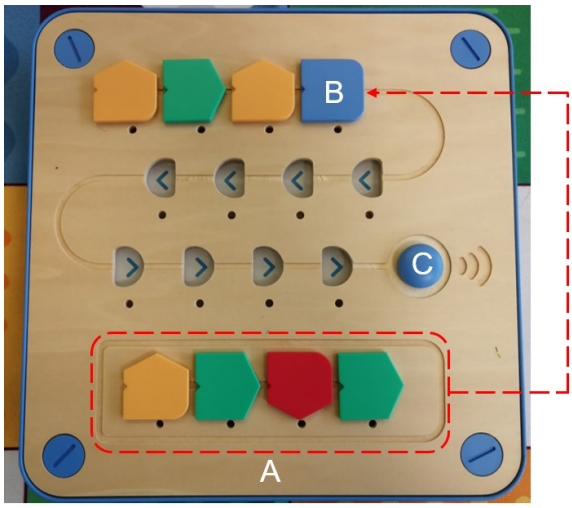

Figure 2. An illustration of the function block of Cubetto [42]**. A small function block A is called in the main function by the blue function tile B in this example.**

Programmed robots or sprites

Most kits enable children to program either physical robots or virtual sprites. Typically, for kits with physical robots, only one robot is programmed each time, while for kits with virtual sprites, several sprites can be programmed at the same time. All of the physical robots move around using a wheel. Some robots borrow the forms of animals, such as Robot Mouse, Bee-Bot, and Blue-Bot. Sensors, lights, and sound effects are embedded in some robots to enrich the robots' interaction with the environment and/or visualize how a robot is executing a program step-by-step, such as Code-a-pillar [15], Electronic Blocks [56], and Thymio Robot.

Most virtual kits are game-based, which require children to create programs to control game sprite(s) to finish certain tasks. For example, LightBot [30] is a coding puzzle game that asks children to program the movement of a virtual robot and light up bulbs. With Move the Turtle, children can program the motion of a turtle sprite to complete different tasks like drawing polygons. A few virtual kits enable other activities like ScratchJr and Codeable Crafts, which support creating interactive stories and animations; and Bitsbox [59], which guides children to create their own apps.

Supporting materials

Kits typically include supporting materials, such as maps, craft materials, and storybooks, to enrich the playing and learning experience for children. For example, Cubetto provides physical adventure maps, which children can use to move around the robot and facilitate their storytelling. Additionally, children can also attach pens to the Cubetto robot and draw its moving tracks, and even decorate the robot with craft materials. ScratchJr [70] and Thymio Robot provide activity, curriculum, and assessment designs for their use in classroom settings for teachers and students. Virtual kits usually have rich scenes and different playing levels for children to engage with, such as Cargo-Bot [60] and The Foos [64].

Perspective 2: Examining computational concepts and practices

One of the main goals of this survey is to examine how existing kits enable young children to explore computational thinking ideas. We use the computational framework from Brennan and Resnick [5] to examine what kinds of computational concepts and practices may be explored by children and how each kit supports that exploration. This framework includes seven computational concepts (Sequences, Events, Parallelism, Loops, Conditionals, Operators, and Data) and four computational practices (being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing), see Table 2 and Table 3.

Computational concepts

Sequences are the most widely supported computational concept among the surveyed kits. Many kits support this concept by enabling children to express their ideas through programming the motion of physical robots or virtual sprites.

A few kits also support Sequences by programming the robots' light and/or sound effects, such as Dash & Dot [55]. Loops are another widely supported computational concept. In tangible programming environments, Loops are realized through encapsulating a sequence of blocks between a start repeat block and an end repeat block, as shown in Figure 3 (a). In graphical programming environments, repeat blocks are typically represented as a "C" shape and repeatedly run the blocks inside this shape, as shown in Figure 3 (b).

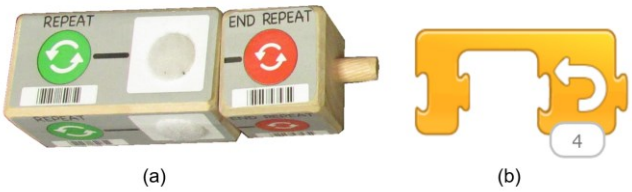

Figure 3. (a) The repeat and end-repeat blocks of KIBO [28]**; (b) The repeat block of ScratchJr** [70]**, which is a rotated "C" shape.**

Events are typically supported by start or trigger buttons. For example, with Cubetto, children can press a blue button (the button C in Figure 2) on the control board to make the robot execute their created program. Some kits also support Events through the interaction between sprites or robots, such as Dash & Dot. The robot Dot can cause robot Dash to move, light up, or play a sound. Most Conditionals are supported in an implicit way, namely that children are not directly exposed to "if-then" statements, but rather program instructions based on different conditions or events. We found three main approaches: (1) sensors are usually embedded in physical robots to sense different environmental conditions so that the robots could be programmed to react accordingly; (2) for physical robots without sensors, maps are usually provided for children to decide how the robots would react to different conditions on a map; and (3) for virtual platforms, a series of commands or special effects like animations, sounds, and light effects could be triggered when sprites interact with other objects on the screen. A few kits also provide "if" blocks with built-in logic blocks to create a conditional code, such as Dr. Wagon as shown in Figure 4.

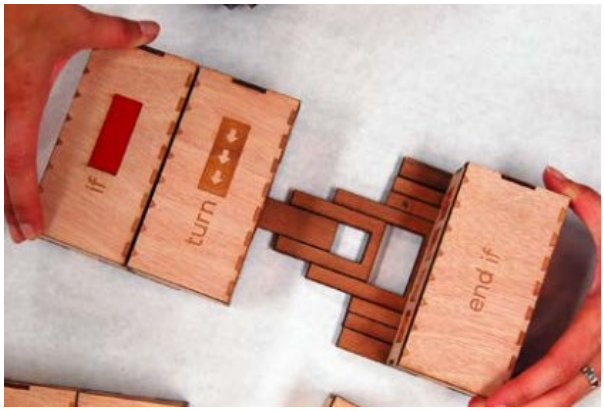

Figure 4. The if blocks of Dr. Wagon [6]**: the left and right ones. In this case, if the robot is above a patch of red, it will turn left.**

Parallelism is primarily supported by virtual and hybrid kits through controlling the motion of several sprites or robots simultaneously or programming the motion, light and sound effects of a sprite at the same time. For example, with Dash & Dot, children can program the robots Dot and Dash to interact with each other; two Curlybots [17] could be synchronized to communicate and run together; and in ScratchJr, several sprites can be programmed to act simultaneously, or a sprite can be programmed to move, play sound, and change size at the same time.

Data is supported by adjusting parameter values such as the motion, rotation, and loop increments. As shown in Figure 1 (b), in Strawbies, a Data block "2" is connected to the "walk" block. Five physical kits in the survey, i.e. Robot Mouse, Bee-Bot, Blue-Bot, Pro-Bot, and Cubetto, implicitly support Data by asking children to repeatedly press the same motion button for a certain number of times, or connecting the same coding blocks together to set parameter values. Operators are the least represented computational concept across all the kits surveyed. In roBlocks, which includes logic operators like "Not", "And", and "Or," when an "And" block is added to the center of two sensors, the corresponding Actuator block can operate only when both sensors are activated. Additionally, Move the Turtle and Puzzlets Starter Pack Game [12] include data manipulation, which involves basic mathematical skills, such as adding and subtracting.

Computational practices

The four computational practices are (1) being incremental and iterative, (2) testing and debugging, (3) reusing and remixing, and (4) abstracting and modularizing. All the surveyed kits support being incremental and iterative by allowing mistakes and supporting an unlimited number of attempts. Children can continuously test their code during the design process to check if their code works properly and debug the problems if something goes wrong.

To assess remixing and reusing, we looked for features that supported children to share and build on others' projects. Most of the surveyed kits do not explicitly support this feature. ScratchJr includes a localized sharing feature that

enables children to share projects across devices. Cargo-Bot, a puzzle game which asks children to program a robot to move crates, allows children to record the playing process and share their solutions on YouTube. Compared to the other three computational practices, relatively few kits support abstracting and modularizing. The ones do support it do so by building a small block of code which can be called in another code sequence. For example, children can build a small function in Cubetto, Light Bot, RoboZZLe, and The Foos. The Digital Dream Lab tabletop puzzle block system also supports the computer science concepts of class and object.

Perspective 3: Examining Expressivity

We define expressivity as the extent to which a kit enables children to explore a range of projects based on their diverse ideas and interests. To examine a kit's expressivity, we reviewed its range of activities and possibilities based on what is claimed by a kit's designers and manufacturers.

Kits range in the level of instructions or pre-determined activities. For example, BitsBox provides step-by-step instructions for children to create a variety of pre-determined games. With Lightbot, children use a subset of coding blocks to program a virtual robot to light up all the blue squares in a scene, as shown in Figure 5. Some kits use narrative as a way to structure children's interactions with computational ideas. In Roberto, children can read a physical storybook and, at different points in the story, program characters using a mobile device and physical programming block stickers [20]; In Hello Ruby, children can follow the adventures of a young girl and her friends. After each chapter, children can engage in different activities that allow them to explore computational concepts and skills [31].

A common activity among many kits is moving a robot or sprite around a space. Some kits aim to constrain a robot's or sprites movement with the goal of engaging children in challenges. In most game-based virtual and hybrid kits like Light Bot, Move the Turtle, Robozzle, and Strawbies, children are tasked with programming sprites to move through a maze or to move within a space to collect items before moving onto other levels. Other kits include resources that facilitate movement within a map of different regions. In Cubetto, children can move a wooden robot within a flat adventure map that has illustrated lakes and grassy areas.

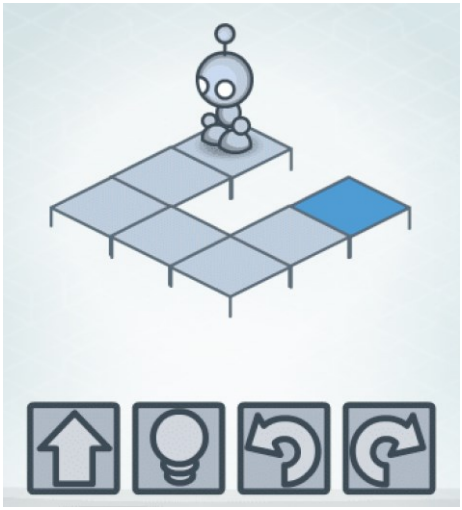

Figure 5. A screenshot of Lightbot game. In this case, players are required to use the bottom four blocks to program the Lighbot to light up the blue square.

Many physical kits support some degree of free exploration, enabling children to decide the routes and destinations for a robot, such as Thymio Robot, Cozmo+Code Lab, and BeeBot. Kits that support storytelling typically allow open-ended exploration of this trajectory. While the Roberto storybook provides a beginning, it asks children to develop their own endings. In ScratchJr and Codeable Crafts, children can create any kind of story. Some kits encourage children to personalize their robots by decorating them with craft materials, such as Cubetto, Dr. Wagon, and meeperBOTS.

While some kits may support individual and group engagement such as KIBO, only a few kits claim to be explicitly designed for group play. For example, Robot Turtles requires at least 2 players at the same time; Hello Ruby includes activities designed to support kids and parents to work together, and Torino [52] supports collaborative learning for children with mixed-visual abilities, i.e. visually-impaired children can play Torino blocks with children without visual impairments.

DISCUSSION

In this section, we discuss design opportunities for computational kits for young children. As we discuss these possibilities, we describe additional kits or technologies that we found inspiring. For every possibility discussed, we need to bear in mind that a kit needs to be developmentally appropriate for young children. We also understand that new possibilities for kits may expand what is considered developmentally appropriate for young children [4].

Expanding how children code

Most of the kits we examined use tangible or graphical blocks to support programming. It would be interesting to explore other forms of programming, such as gesture, body movement, and sound. One example is the video sensing feature in Scratch [25]. Children can create projects in Scratch that use a computer's camera to simulate object and

gesture recognition. For example, the Scratch project Simple Bubbles [8] uses the camera to capture hand movements as the input to pop the bubbles. Even though video sensing is an input method rather than a programming method in this example, such interaction can inspire new programming methods.

Additionally, many of the surveyed physical and hybrid kits have separate programming blocks and robots or sprites, and most of their programming block designs adopt either tile or cube forms. While a few kits have different designs of coding blocks, such as the embedded coding buttons on Blue-Bot, it would be interesting to explore how coding blocks could be designed in other forms, particularly beyond tiles or cubes. For example, Ozobot robot [71] allows children to program its motion by drawing lines of different colors. Tangible Programming with Train [35] uses code bricks which are also part of the train track to program a train's motion. Instead of having separated coding blocks and controlling robots, the programming blocks of roBlocks [47] can be stacked together to create a robot which is also controlled by these blocks.

Expanding what children can code

In a paper reflecting on the changing nature of children's programming, Michael Eisenberg and colleagues discussed expanding the possibilities to include more programmable materials, settings, and surfaces [14]. These new possibilities included embedding computing in everyday materials and objects like paper, textiles, and other media to enable people to engage in computing in more familiar and natural ways. With Chibitronics Love to Code [72], children can build interactive paper circuits and program LEDs, using a microcontroller called the Chibi Chip. Proctor and colleagues built an interactive fiction web application that combined textual literacy and computational literacy [43]. By incorporating computational concepts like sequences, loops, and conditionals into interactive fiction games, readers can choose how storylines develop and unfold different story experiences. We also see possibilities to explore other programmable objects or phenomena, such as light, sound, or even people. One such example is Unruly Splats [2], where children can program a physical tile to light up or turn off when touched. As children interact with their programmed tiles, children can run around and jump on the tiles to play active games, essentially programming children's body movements. With COJI robot, children can program the emotion expression on the screen of the robot. Music Blocks [35] uses sound as a programming medium, with which children can create a melody by inserting cubical blocks that represent different musical phrase into a music player.

For virtual and hybrid kits that include one or more sprites, the user interfaces are highly homogeneous, i.e. either through a PC screen or a mobile-device screen. It would be interesting to explore other forms of interfaces that can enrich children's interactions with computational kits, such as holographic projection, virtual reality (VR), and augmented reality (AR). These technologies have already been deployed to promote children's learning and playing. In the Virtual Environment Interactions project, young people firstly learned dance moves, then programmed similar dance moves for characters in VR [10]. An AR-based video-modeling storybook was developed to help children with autism improve their perceptions and judgments of facial expressions and emotions [7].

Finally, one of the construction kit principles proposed by Resnick and Silverman encourage designing for "wide walls", or enabling children to explore many different projects to represent their different interests [42]. We see the potential for kits to expand their expressivity, or the range of activities that children can explore and engage in. While step-by-step instructions or specific challenges can help to scaffold children's learning experiences, more open-ended systems can enable children to express ideas, develop their own goals, and pursue more personally meaningful projects.

Expanding who can code

Most of the surveyed kits, especially the virtual kits, are optimized for individual engagement. While some kits support sharing projects across a local network of mobile devices, an online community can expand who children can learn with. In the Scratch online community, children can get feedback, work on projects together, and learn from other creators [46]. Remixing, or building on others' project can be a meaningful pathway to support computational thinking [11].

In addition to expanding children's networks, computational kits could also consider more explicitly the kinds of roles that parents or other adult caregivers can play. Hello Ruby, for example, designed activities with children and adult caretakers in mind. Online platforms could also create spaces for parents and adult caregivers to share their experiences in working with their children and learn about more opportunities for their families.

Finally, we see broadening participation in computing as an important consideration when exploring new possibilities for kits. We especially want to highlight the opportunities for kits to support children with disabilities, such as children with visual, hearing, and physical disabilities. Among the surveyed kits, only Torino is specifically designed to help visually-impaired children learn computational concepts and skills. We suggest that more kits be designed to meet the needs of children with disabilities. Some researchers have suggested design principles and strategies for designing interactive technologies for children with disabilities. For example, Meryl Alper and colleagues [1] suggested that, in addition to the three dimensions— "low floors", "high ceilings", and "wide walls"—proposed by Resnick and Silverman [45], designers should consider the fourth dimension of "reinforced corners", namely to support exceptional children who can thrive at the widest walls, highest ceilings, and lowest floors. Juan Pablo Hourcade

summarized suggestions for designing participatory technologies and activities for children with autism, such as engaging children deeply, involving stakeholders, and designing the ecology around the technology [23]. These strategies can serve as guiding principles for designing computational kits for young children with disabilities.

Expanding what children can explore

Some of the computational concepts such as Sequences, Loops, Events, and Conditionals are well supported by existing kits. More specifically, Sequences are supported by programming a series of instructions for a robot or sprite; Loops are supported through repeat blocks that can execute a block of code repeatedly; Events are achieved by a physical or graphical start button that triggers the movement of programmed robots or sprites; Conditionals are supported through programming how sprites or robots react to certain conditions, which are typically in response to sensors or adventure maps; Data is primarily supported by adjusting the values of a parameter, such as the distance in motion or how many times a loop repeats; Parallelism can be achieved through programming more than one robot or sprite within a kit at the same time, or coordinating the motion, light, and sound effects of the kit simultaneously.

It would be interesting to explore how these computational concepts can be supported through other approaches. For example, how could visual, auditory, or tactile objects be programmed to support Sequences and Loops in addition to a robot's or sprite's motion? Using BlockyTalky [49], children can create interactive music systems or digital instruments while exploring concepts such as distributed systems. For Events, what other kinds of actions, beyond a start and stop button, can children engage in to trigger actions? Depth and 3D sensors like the XBox Kinect [73] can open up possibilities for people to use body movements or gestures to trigger events.

While a computational kit does not necessarily need to support most computational concepts and practices, there are opportunities for exploring the less prevalent computational concepts and practices, such as Operators. It would be interesting to explore the possibilities with logical operations such as "And" and "Or". While many kits support computational practices like experimenting and iterating, we see potential to more explicitly support practices such as debugging and abstraction. For example, Robo-blocks [39] is designed to support debugging activities for early primary school children through the use of debugging flags to identify and mark coding problems. As we explore more ways to support children to explore computational concepts and practices, we can open up more opportunities for children to create, invent, and express themselves in new and empowering ways.

CONCLUSION

Cultivating computational thinking can enable young children to engage with powerful ideas, to express themselves in new ways, and to understand the changing and increasingly digital world around them. Our survey provides an overview of computational kits for young children (ages 7 and under) that have emerged across academia and industry. The findings reveal the commonalities across existing kits and highlight ways for designers and researchers to expand the possibilities for children to create, explore, and play with computing. The surveyed kits span across physical, virtual, and hybrid kits and were examined across three perspectives: design features, how children could explore computational concepts and practices, and what range of activities or projects children could engage in.

When examining their design features, kits often used blocks or puzzle-pieces to represent code that controls robots or virtual sprites. The most common computational concepts that children could explore were sequences, loops, events, and conditionals. Finally, kits ranged in the activities and projects they enabled. Some were constrained to specific challenges such as moving through a maze, while some provided some scaffolds such as a narrative to structure the experience. Relatively few kits enabled more open-ended engagement and creation.

Through our survey, we see possibilities for expanding how children can code, what they can code, who can code, and what they can explore. These possibilities include new modes of expression such as body motion or new media such as light and sound. We also see possibilities for who designers can better support, such as more explicit roles for adult caregivers and expanding possibilities for children with disabilities. Finally, while many kits enable the exploration of some computational concepts and practices, we see opportunities to expand how these concepts are supported as well as new concepts they could explore.

ACKNOWLEDGMENTS

The authors thank Marissa Wajda for helping get the toys and kits survey started, Fujiko Robledo Yamamoto for reviewing the paper, the anonymous reviewers for helpful feedback, and colleagues such as Victor Lee and Michael Horn who shared their in-progress work with us on computational kits for children.

SELECTION AND PARTICIPATION OF CHILDREN

No children participated in this work.

REFERENCES

1. Meryl Alper, Juan Pablo Hourcade, and Shuli Gilutz. 2012. Adding Reinforced Corners: Designing Interactive Technologies for Children with Disabilities. *Interactions* 19, 6: 72–75. https://doi.org/10.1145/2377783.2377798
2. David Kunitz Amon Millner, Bryanne Leeming, Paayal Khanna, Daniel Ozick. Unruly Splats. Retrieved from https://www.kickstarter.com/projects/bryanneleeming/unruly-splats-active-stem-play
3. Marina Umaschi Bers. 2017. *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.

4. Marina Umaschi Bers and Michael S Horn. 2010. Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. *High-tech tots: Childhood in a digital world*: 1–32.
5. Karen Brennan and Mitchel Resnick. 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
6. K Chawla, M Chiou, A Sandes, and P Blikstein. 2013. Dr. Wagon: A "stretchable" toolkit for tangible computer programming. *ACM International Conference Proceeding Series*: 561–564. https://doi.org/10.1145/2485760.2485865
7. Chien Hsu Chen, I. Jui Lee, and Ling Yi Lin. 2016. Augmented reality-based video-modeling storybook of nonverbal facial cues for children with autism spectrum disorder to improve their perceptions and judgments of facial expressions and emotions. *Computers in Human Behavior* 55: 477–485. https://doi.org/10.1016/j.chb.2015.09.033
8. Chrisg. Simple Bubbles. Retrieved from https://scratch.mit.edu/projects/10005522/
9. Douglas H Clements. 1984. Effects of Computer Programming on Young Children's Cognition Effects of Computer Programming on Young Children's Cognition. 76, NOVEMBER 1984: 1051–1058. https://doi.org/10.1037/0022-0663.76.6.1051
10. Shaundra B. Daily, Alison E. Leonard, Sophie Jörg, Sabarish Babu, and Kara Gundersen. 2014. Dancing Alice: exploring embodied pedagogical strategies for learning computational thinking. *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14*: 91–96. https://doi.org/10.1145/2538862.2538917
11. Sayamindu Dasgupta, William Hale, Andrés Monroy-Hernández, and Benjamin Mako Hill. 2016. Remixing as a Pathway to Computational Thinking. 1438–1449. https://doi.org/10.1145/2818048.2819984
12. Digital Dream Lab. Puzzlets. Retrieved from http://www.digitaldreamlabs.com/
13. H Ehsan, C Beebe, and M E Cardella. 2017. Promoting computational thinking in children using apps. *ASEE Annual Conference and Exposition, Conference Proceedings* 2017–June. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85030538569&partnerID=40&md5=fd46714b61a8323926a755f4bace060d
14. Michael Eisenberg, Nwanua Elumeze, Michael MacFerrin, and Leah Buechley. 2009. Children's programming reconsidered. *Proceedings of the 8th International Conference on Interaction Design and Children - IDC '09*: 1. https://doi.org/10.1145/1551788.1551790
15. Fisher-Price. Think & Learn Code-a-pillar. Retrieved from http://fisher-price.mattel.com/shop/en-us/fp/think-learn-code-a-pillar-dkt39
16. Louise P. Flannery, Elizabeth R. Kazakoff, Paula Bontá, Brian Silverman, Marina Umaschi Bers, Mitchel Resnick, Elizabeth R. Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: Support for Early Childhood Learning Through Computer Programming. *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*: 1–10. https://doi.org/10.1145/2485760.2485785
17. P. Frei, V. Su, B. Mikhak, and H. Ishii. 2000. Curlybot: designing a new class of computational toys. *Proceedings of the SIGCHI conference on Human factors in computing systems* 2, 1: 129–136. https://doi.org/http://doi.acm.org/10.1145/332040.332416
18. Google. Blocky Games. Retrieved from https://blockly-games.appspot.com/?lang=en
19. Michael S. Horn, R. Jordan Crouser, and Marina U. Bers. 2012. Tangible interaction and learning: The case for a hybrid approach. *Personal and Ubiquitous Computing* 16, 4: 379–389. https://doi.org/10.1007/s00779-011-0404-2
20. Michael S Horn, Sarah Alsulaiman, and Jaime Koh. 2013. Translating Roberto to Omar : Computational Literacy, Stickerbooks, and Cultural Forms. *IDC 2013*: 120–127. https://doi.org/10.1145/2485760.2485773
21. Michael S Horn, Erin Treacy Solovey, R Jordan Crouser, and Robert J K Jacob. 2009. Comparing the use of tangible and graphical programming languages for informal science education. *Proceedings of the 27th international conference on Human factors in computing systems CHI 09* 32: 975. https://doi.org/10.1145/1518701.1518851
22. Michael S Horn, Erin Treacy Solovey, and Robert J.K Jacob. 2008. Tangible Programming and Informal Science Learning: Making TUIs Work for Museums. *IDC '08 Proceedings of the 7th international conference on Interaction design and children*: 194–201. https://doi.org/10.1145/1463689.1463756
23. Juan Pablo Hourcade. 2017. PARTICIPATORY DESIGN WITH CHILDREN IN THE AUTISM SPECTRUM. *Participatory Design for Learning: Perspectives from Practice and Research*: 111.
24. Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. 2015. Strawbies: Explorations in Tangible Programming. *Proceedings of the 14th International Conference on Interaction Design and Children - IDC '15*: 410–413. https://doi.org/10.1145/2771839.2771866
25. Ting-hsiang Tony Hwang. 2012. Exploring Real-time

Video Interactivity with Scratch. Massachusetts Institute of Technology.

26. INSPIRE Research Institute for Pre-College Engineering. 2017. 2017 Engineering Gift Guide. Retrieved from https://engineering.purdue.edu/INSPIRE/EngineeringGiftGuide
27. Filiz Kalelioğlu, Yasemin Gülbahar, and Volkan Kukul. 2016. A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic J . Modern Computing* 4, 3: 583–596.
28. KinderLab Robotics. KIBO. Retrieved from http://kinderlabrobotics.com/kibo/
29. Learning Resources. Robot Mouse. Retrieved from https://www.learningresources.com/product/learning+essentials--8482-+stem+robot+mouse+coding+activity+set.do
30. LightBot Inc. Lightbot. Retrieved from http://lightbot.com/index.html
31. Linda Liukas. Hello Ruby. Retrieved from https://www.kickstarter.com/projects/lindaliukas/hello-ruby
32. Caitlin K Martin, Nichole Pinkard, Sheena Erete, and Jim Sandherr. 2016. Connections at the Family Level: Supporting Parents and. In *Moving Students of Color from Consumers to Producers of Technology*. IGI Global, 220–244.
33. Dana Charles McCoy, Hirokazu Yoshikawa, Kathleen M. Ziol-Guest, Greg J. Duncan, Holly S. Schindler, Katherine Magnuson, Rui Yang, Andrew Koepp, and Jack P. Shonkoff. 2017. Impacts of Early Childhood Education on Medium- and Long-Term Educational Outcomes. *Educational Researcher* 46, 8: 474–487. https://doi.org/10.3102/0013189X17737739
34. Timothy S. McNerney. 2000. Tangible Programming Bricks : An approach to making programming accessible to everyone. *Media*, June 1983.
35. Timothy S. McNerney. 2004. From turtles to Tangible Programming Bricks: Explorations in physical language design. *Personal and Ubiquitous Computing* 8, 5: 326–337. https://doi.org/10.1007/s00779-004-0295-6
36. Modular Robotics. Cubelets. Retrieved from https://www.modrobotics.com/cubelets/
37. National Research Council. 2011. *Report of a workshop on the pedagogical aspects of computational thinking*. National Academies Press.
38. Next is Great. Move the Turtle. Retrieved from http://movetheturtle.com/
39. N Nusen and A Sipitakiat. 2012. Robo-blocks: a tangible programming system with debugging for children. *IDC '12 Proceedings of the 11th International Conference on Interaction Design and Children*: 98–105. https://doi.org/10.1145/2307096.2307108
40. Hyunjoo Oh, Anisha Deshmane, Feiran Li, and JY Han. 2013. The digital dream lab: tabletop puzzle blocks for exploring programmatic concepts. *Tei 2013*: 51–56. https://doi.org/10.1145/2460625.2460633
41. Plobot Team. Plobot. Retrieved from http://plobot.com/#home
42. Primo. Cubetto. Retrieved from https://www.primotoys.com/
43. Chris Proctor. 2017. Interactive fiction : Weaving together literacies of text and code. 555–560. https://doi.org/10.1145/3078072.3084324
44. Hayes Solos Raffle, Amanda J Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. *System* 6, 1: 647–654. https://doi.org/10.1145/985692.985774
45. Mitchel Resnick and Brian Silverman. 2005. Some Reflections on Designing Construction Kits for Kids. *Proceeding of the 2005 conference on Interaction design and children (IDC '05)*: 117–122. https://doi.org/10.1145/1109540.1109556
46. Natalie Rusk. 2016. Makeology: Identities, Materials, and Educational Outcomes. In *Makeology: Makerspaces as Learning Environments*, Y. K. Peppler, E. Halverson, Kafai (ed.).
47. Eric Schweikardt and Mark D Gross. 2006. roBlocks: a robotic construction kit for mathematics and science education. *Proceedings of the 8th international conference on Multimodal interfaces*: 72–75. https://doi.org/10.1145/1180995.1181010
48. Dan Shapiro. Robot Turtles. *Thinkfun*. Retrieved from http://www.robotturtles.com/
49. R Benjamin Shapiro, Rebecca Fiebrink, Matthew Ahrens, and Annie Kelly. 2016. BlockyTalky: A Physical and Distributed Computer Music Toolkit for Kids. *Proceedings of the International Conference on New Interfaces for Musical Expression* 16: 427–432. Retrieved from http://www.nime.org/proceedings/2016/nime2016_paper0084.pdf
50. Amanda Sullivan, Mollie Elkin, and Marina Umaschi Bers. 2015. KIBO Robot Demo: Engaging Young Children in Programming and Engineering. *Proceedings of the 14th International Conference on Interaction Design and Children - IDC '15*: 418–421. https://doi.org/10.1145/2771839.2771868
51. Target. Coding Toys. Retrieved from https://www.target.com/c/coding/-/N-4sj2p
52. Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. 2017. Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments.

Proceedings of the 2017 Conference on Designing Interactive Systems - DIS '17: 739–752. https://doi.org/10.1145/3064663.3064689

53. TTS Group. Bee Bot. Retrieved from http://www.tts-group.co.uk/bee-bot-rechargeable-floor-robot/1001794.html
54. J M Wing. 2006. Computational Thinking. *Communications of the ACM*.
55. Wonder Workshop. Dash & Dot. Retrieved from https://www.makewonder.com/
56. P. Wyeth and Gordon Wyeth. 2001. Electronic blocks: Tangible programming elements for preschoolers. *Proceedings of the Eighth IFIP TC13 Conference on Human-Computer Interaction*: 496–503. Retrieved from http://archive.itee.uq.edu.au/~peta/WyethInteract.pdf
57. Oren Zuckerman and Mitchel Resnick. 2003. A physical interface for system dynamics simulation. *CHI '03 extended abstracts on Human factors in computer systems - CHI '03*: 810. https://doi.org/10.1145/765999.766005
58. Pro-Bot. Retrieved from https://www.bee-bot.us/probot.html
59. Bits Box. Retrieved from https://bitsbox.com/
60. Cargo-Bot. Retrieved from https://twolivesleft.com/CargoBot/
61. Codeable Crafts. Retrieved from https://www.codeablecrafts.com/
62. RoboZZle. Retrieved from http://www.robozzle.com/
63. Run Marco! Retrieved from https://www.brainpop.com/games/runmarco/
64. The Foos. Retrieved from https://codespark.com/webgl/
65. COZMO + Code Lab. Retrieved from https://www.anki.com/en-us/cozmo/code-lab
66. Thymio Robot. Retrieved from https://www.thymio.org/en:thymio
67. meeperBOTS. Retrieved from https://meeperbot.com/pages/meeperbots
68. COJI. Retrieved from https://wowwee.com/coji
69. Blue-Bot. Retrieved from https://www.bee-bot.us/bluebot.html
70. ScratchJr. Retrieved from http://scratchjr.org
71. Ozobot. Retrieved from https://ozobot.com/
72. Chibitronics Love to Code: Chibi Chip & Cable. Retrieved from https://chibitronics.com/shop/love-to-code-chibi-chip-cable/
73. Kinect. Retrieved from https://www.xbox.com/en-US/xbox-one/accessories/kinect