A Survey of Interface Representations in Visual Programming Language Environments for Children's Physical Computing Kits

Sarah Brown
Embodied Learning and Experience Lab, University of Florida
Gainesville, USA
sarah.brown@ufl.edu

Sharon Lynn Chu
Embodied Learning and Experience Lab, University of Florida
Gainesville, USA
slchu@ufl.edu

Pengfei Yin
Embodied Learning and Experience Lab, University of Florida
Gainesville, USA
pengfeiyin@ufl.edu

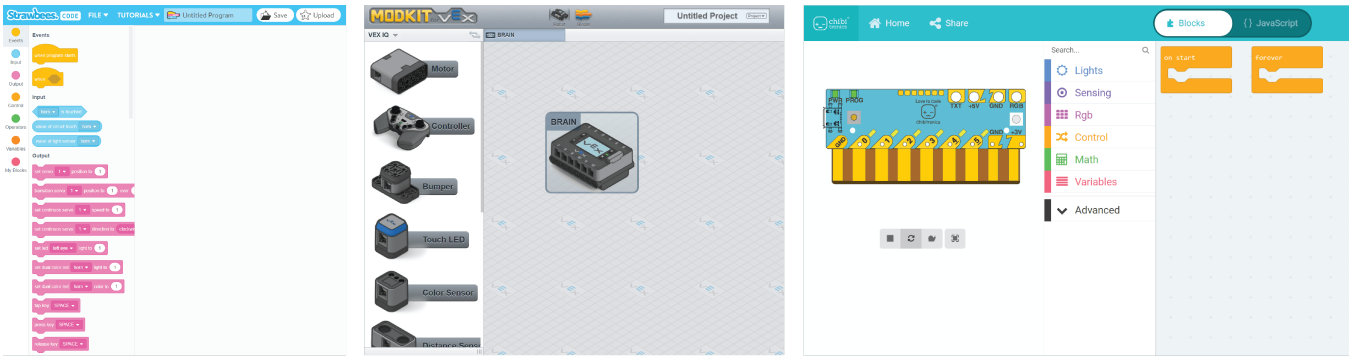

Figure 1: Examples of Visual Programming Languages for Physical Computing Kits: Strawbees Code [1], Modkit [25], and MakeCode Chibitronics [33]

ABSTRACT

Physical computing toolkits for children expose young minds to the concepts of computing and electronics within a target activity. To this end, these kits usually make use of a custom Visual Programming Language (or VPL) environment that extends past the functionality of simply programming, often also incorporating representations of electronics aspects in the interface. These representations of the electronics function as a scaffold to help the child focus on programming, instead of having to handle both the programming and details of the electronics at the same time. This paper presents a review of existing physical computing toolkits, looking at the What, How, and Where of electronics representations in their VPL interfaces. We then discuss potential research directions for the design of VPL interfaces for physical computing toolkits for children.

CCS CONCEPTS

• **Human-centered computing** → **Graphical user interfaces**.

KEYWORDS

physical computing kits; visual programming languages for children; electronics; robots

ACM Reference Format:
Sarah Brown, Sharon Lynn Chu, and Pengfei Yin. 2021. A Survey of Interface Representations in Visual Programming Language Environments for Children's Physical Computing Kits. In *Interaction Design and Children (IDC '21), June 24–30, 2021, Athens, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3459990.3460727

IDC '21, June 24–30, 2021, Athens, Greece

ACM ISBN 978-1-4503-8452-0/21/06…$15.00
https://doi.org/10.1145/3459990.3460727

1 INTRODUCTION

There have been a great many physical computing toolkits developed for children, particularly in the past 30 years [7]. These incorporate some form of physical construction with the ability to program one's creation. Physical computing kits range from programmable robots like the Thymio bot [40], to micro-controller-based solutions like Talkoo, which implements the use of an Arduino [21]. The benefits of these kits and their accompanying VPLs are clear: they provide a multi-disciplinary learning experience for the child, which encourages them to construct new understanding of concepts from their own experiences [8]. Through this process, physical computing kits have been shown to increase interest in programming and STEM activities, as well as students' overall enjoyment and engagement [6, 39].

Many of these kits and toys come with a custom Visual Programming Language, or VPL, which is used by the child to program the accompanying hardware towards some sort of educational benefit. However, these unique programming environments often surpass the functionality of a traditional language or VPL interface, enabling children to engage with computational hardware such as

electronic components as well. Due to the multi-disciplinary nature of these kits and toys, it is common then to find representations of the electronics embedded in the custom VPL interface itself, serving to scaffold understanding between the computation and electronics being programmed. We provide a handful of examples of these physical computing VPL interfaces in Fig 1 [1, 25, 33] that contain aspects of the electronics that children can program through the programming interface. The embeddedness of interface representations of the electronic components in the VPL typically helps to visually promote the connections between the electronics and coded instructions, as a form of scaffold for the child during programming.

However, to the best of our knowledge, a comprehensive review has not been conducted of the ways in which these VPLs for physical computing kits scaffold understanding of electronic components within their respective programming environments. To provide a foundation for the understanding of how to design these VPLs and identify future avenues for research, we present an analysis of their design, examining the ways they represent electronics in terms of What is represented, How it is represented, and Where it is represented.

1.1 Goals of Review

The goal of this review is to identify common design strategies for representing electronic components in the user interfaces of Visual Programming Languages for children within the context of physical computing kits. This understanding can then aid in informing future designs and investigations of these specialized programming environments for children.

2 BACKGROUND AND RELATED WORK

2.1 Physical Computing

As summarized by Blikstein [8], physical computing kits were born from Papert's concept of *constructionism*, which proposes that (based on Piaget's theory of *constructivism*) learners are able to understand new theories about their world if they construct these new theories from their own experiences, and that the best way of doing this is through the creation of a shareable object by the learner. The nature of these shareable objects have evolved greatly over the years. While early examples of physical computing kits included programmable bricks, such as the Lego Mindstorms RCX [22], these kits have since taken many forms, using a variety of micro-controllers (such as the Arduino or BBC Microbit [9]), and non-electrical components (such as textiles [12] or toys [24]). However, the general premise is unanimous: learning is more engaging when one interacts with and programs tools and materials to create a computational artifact. It is an embodied experience, explained by Katterfeldt and colleagues as 'Begreifbarkeit' [22]. Derived from the German word for 'graspable,' they use this term to explain the way the body and mind work together to achieve understanding. It calls for more than the mere creation of an artifact, but for iterative problem-solving and the understanding that comes through such a process. In the end, this understanding is not limited to just one discipline. Children come to learn about construction, robotics, electronics, science, and computation by engaging in these multi-disciplinary experiences [18, 38], while also increasing their interest and engagement in STEM topics [6, 39].

2.2 Visual Programming Languages for Children

The idea of a programming interface designed specifically for children is far from new, and in recent years many physical computing kits have been accompanied by custom VPLs made to be more accessible to children. An early example of a VPL created with children in mind is Kahn's ToonTalk [20], which sought to teach formal programming concepts in a fun and accessible way. Their method of choice was the use of visual metaphor - every computational component was made concrete as some sort of graphic or animation. Its goals were twofold: to be easy for children to learn while also providing a powerful programming environment. Since then, many VPLs for children have followed in its footsteps, varying between the degrees of learning ease and capabilities as a programming environment. There is a children's VPL for every occasion - while examples such as Scratch, a VPL which allows for the creation of rich media projects [28], offer great flexibility, others aim for a younger audience, such as the VPL that accompanies Terrapin's Blue-Bot [47], which allows the user to send simple instructions to a pre-assembled robot.

The interfaces of these visual languages for children rely heavily on the use of blocks [55], which come embedded with computational instructions that are then stacked together to form a sequence of code. Block-based programming languages have been widely used in teaching children computational concepts prior to learning to code, and it has been found that children who have block-based programming experience perform better when introduced to text-based programming compared to their peers with no experience [17]. This makes VPLs a natural fit for physical computing kits, providing a low barrier to entry into the programming of their constructed artifacts.

Such VPLs are typically packaged with an accompanying electronic component(s) - such as a robot, LEDs, or micro-controller. These electronic components are often embedded within the VPL interface itself. For example, the event blocks in the Thymio VPL interface intentionally mimic the design of the Thymio robot, providing a direct, visual mapping between the computation and the electronics that it impacts [43, 49] (Figure 2). This mapping between the digital representation in the VPL interface and the real-world physical component allows children to create quicker connections between how the code they are creating will affect their constructed artifact. Other examples of interface representations of the physical components of a kit include showing icons of the hardware on the computing blocks themselves [26], portraying simulations of the electronics [18, 35], or having a block that is composed of a picture of the electronic component itself [25].

2.3 Interface Representations of Physical Components as Scaffolds

Scaffolding, as described by the broader educational literature, is the process through which an expert intervenes in the learning process of the novice [29]. This can manifest in a variety of approaches, including demonstrating solutions to challenges the learner faces,

Figure 2: An example of coding blocks in the Thymio bot's VPL (left) next to an image of the Thymio bot (right).

maintaining the direction or focus of the learner, or simplifying the nature of a task [3]. In the case of VPLs in physical computing kits, we suggest that the designer scaffolds understanding for the child user through design elements which elucidate the connections between the in-the-screen computation and the in-the-world electronics being programmed. As described early on by Wood et al. [52], this is well aligned with the purpose of scaffolding to allow novices, such as children, to solve problems that they would be unable to without assistance. A newcomer to physical computing may not implicitly understand the ways the code they create impacts the physical artifact, or be able to imagine all the different possibilities that the code enables for the artifact. With physical objects, one can easily tinker and experiment quickly with an artifact during its creation. For example, a sculptor moulding clay to create a sculpture does so by interacting directly with the clay in different ways. However, for the creation of computational physical artifacts, one has to interact with code that then controls the artifact. This two-step process for creation may easily create a cognitive burden on a learner, resulting in perhaps poorer creations or less learning. Thus, in physical computing interfaces, since the expert is not physically present, the scaffolds they provide to children are embedded in the design of the interface itself, and serve to guide them through the learning process.

2.4 Existing Surveys of Physical Computing Kits

Prior reviews in this area tend to focus on reviewing the physical computing kits as a whole, or on the educational outcomes of these kits. For example, Blikstein's review of physical computing toolkits provides an in-depth history of how these constructionist kits have evolved over time [7]. In his discussion of today's design imperatives, he stresses the importance of investigating the design of computing experiences for a wide range of learners. Yu and Roque's survey of computational toys and kits for children gives further insight into how existing kits provide these computing experiences. They noted that all computational environments included with these artifacts utilized blocks, either digital or physical in the case of tangible computing examples [55]. They suggested these blocks have the potential to expand past the typical tile-like shapes, or could be in close proximity to the hardware such as in roBlocks [41], where the tangible programming blocks form the robot itself. This is another form of scaffolding understanding between the electronics and computation, but rather than achieving this scaffolding through visual representations, it is achieved through a tangible joining of the computation with the electronic assembly.

In their 2019 review, Yu and Roque examined a variety of computing kits for their ability to support computational concepts [56], finding that many supported the concepts of loops, sequencing, conditionals, and data. This is supported by Sullivan and Heffernan's review of robotics construction kits [45], who noted as a key finding that these kits support a progression of learning computational thinking, starting with the concept of sequencing. They also noted that the kits they surveyed provided both a robotics education as well as supported understanding in other domains. In a similar strain of inquiry, Lye and Koh [27] surveyed ways in which computational thinking is incorporated into K-12 curricula. Their implications included the need for more *"constructionism-based problem solving learning environment(s) (PSLE),"* whose approaches are backed by evidence. Most constructionism-grounded kits discussed rely on children experiencing and interacting with real-world materials, including pre-assembled toys or electronics that are not pre-assembled for use. We did not find any existing reviews that focus on analyzing user interfaces of VPLs in children's physical computing kits. Understanding how specific aspects of the interfaces are designed can reveal what has been attempted and what has not, and thus point to future design directions.

3 PAPER SELECTION

This review includes both academic papers and commercial examples of physical computing kits and programmable robots. Searches were ran on a web search engine, and a researcher perused each link on the first 5 pages of the search results pages. Some of these kits were associated with research papers, and others were commercial in nature. In total, 45 physical computing kits (28 commercial and 17 research-based) with VPLs were found. The following criteria were then applied to all the kits found to exclude those deemed not relevant for our review. To be included, the kit or toy had to include a physical component (such as a robot or electronics to assemble), and a VPL with which to program said component. This was to ensure our focus on physical computing activities that included a VPL for us to examine. Furthermore, we only included kits that used a custom-made VPL instead of a generic programming application, such as the Arduino Editor. No other criteria were used. In the end, 30 kits (26 commercial and 4 research-based) were kept for inclusion in the analysis.

4 ANALYSIS OF PHYSICAL COMPUTING KIT VPL INTERFACES

The focus of our analysis was on representations of electronics in the custom VPL interfaces of the physical computing kits found. We reviewed how electronics were represented within these VPLs across three dimensions: (i) *What* was represented – referring to what specific electronics components were represented in the VPL; (ii) *How* it was represented – the visual method used to represent said electronics; and (iii) *Where* it was represented – where in the VPL the electronics were represented, relative to common components of VPLs, such as the programming blocks and programming canvas. This framework intends to capture the design decisions made in the creation of these VPLs as they scaffold understanding

between electronic components and the programming interface, by carefully examining what kinds of representations are present, and where and how they are represented in the VPL itself. Two researchers conducted the analysis to extract information for these dimensions for each VPL included in the review, either through descriptions of the VPL (which were commonly provided in our research-based examples, such as academic papers) or through live versions of the VPL (which were commonly found in our commercial examples). Afterwards, the researchers went through the information and performed a qualitative open coding process, assigning descriptive codes to the information extracted. It was possible for a single dimension of a VPL to contain multiple codes. One researcher then collapsed all the codes into a final coding scheme. The codes were then compiled by count, and summary statistics were produced.

5 RESULTS

We present our review results by each of the three dimensions analyzed. The summary statistics of the codes generated for each dimension are shown in Table 1, and examples of each code are shown in Figure 3. A full breakdown of the codes assigned to each kit/VPL is provided in Table 2.

5.1 *What* Electronics Are Represented

Four codes were found for the *What* dimension, which described what electronics the designers of the surveyed VPLs decided to represent in these interfaces. **Individual Components** was the most prevalent code (46.46%) and was characterized by a representation of actual hardware components, such as motors or LEDs, in the VPL interface. Our provided example in Figure 3 is a block from Strawbees Code [1], which represents a light sensor as an icon and text on a block. The code **Micro-controller** indicated representations of a micro-controller in the VPL interface, such as an Arduino or BBC Microbit. The example representation of a micro-controller shown in Figure 3 is from Cabrera et al.'s research [9, 34], which used the VPL MakeCode, which included a live simulation of the BBC Microbit controller. Lastly, the code **Robot** was for any sort of assembled robot that was represented in the VPL. An example of this would be the small icons of the Meeperbot shown in their mobile VPL [30]. And finally, **N/A** indicated that no electronics where represented in the VPL, and is used the same for *How* and *Where*.

5.2 *How* Electronics Are Represented

Five codes were found for the *How* dimension, which described how the chosen electronics were represented in the VPLs. The most prevalent of these, after N/A (35.48%) was **Text** (32.36%), which indicates that the electronics were represented in a text format, perhaps the simplest option to represent the electronics within the programming environment. An example of this would be the plain text descriptions of electronics found on the blocks in Pico Cricket's VPL [11]. The code **Simulation** refers to the electronics being represented in the form of a simulation within the VPL interface. In Seyed et al.'s work, they used the simulation of an arcade machine in Microsoft's MakeCode Arcade in conjunction with physical computing hardware that allowed the students to assemble small arcade machines [35, 42]. This allowed students to see very easily how the changes in their code affected the electronics, providing scaffolding through a virtual representation of the computational tasks the students coded. VPLs who received the code **Icons** in this dimension represented the electronics as simple graphical icons, unlike **Image** which refers to electronics being represented as a photographic or complex graphical image. Our only example of **Image** is the Modkit hardware blocks [25], while an example of **Icons** would be Grasp IO's blocks [26], which uses simple icons to represent hardware components such as a servo. These kinds of graphical representations do not go as far as simulations do in scaffolding understanding between the electronics and the computation. However they still bring the electronics to the same graphical level as the programming blocks, allowing students to associate the two together and make appropriate connections.

5.3 *Where* Electronics are Represented

Four codes were found for the *Where* dimension. The most prevalent code after N/A (40.00%) was **In Blocks** (26.67%), where the electronics were represented in the computational blocks used to construct programs within the VPL, creating a direct connection between the implied piece of computation the block represented and the electronic it affects. In the commercially available HoneyComb Queen Kit, the block-based VPL includes textual representations of hardware components such as LEDs on the blocks themselves [14]. **Separate Blocks** indicated that the electronic representations were treated as the entire programming block within the VPL. The VPL developed for the Thymio robot uses the entire block to display an iconic representation of the Thymio robot, thus qualifying it as representing the electronics as a separate block of its own [48]. This method of scaffolding has different implications - here, the user might think more holistically about the electronics as they program, as the computation is represented alongside a full representation of the electronics (in the case of the Thymio robot). Additionally, perhaps this approach where the electronics surrounds the computation, alters which the child considers first in placing the block - the computation or the electronics, as opposed to when the computation surrounds the electronics. Lastly, the code **Separate UI** refers to when electronics were displayed in their own section of the VPL, outside of the canvas area used to program. For example, in Microsoft's MakeCode Chibitronics, the Chibitronics micro-controller is displayed to the left of the programming canvas, in its own UI space [33]. This approach, like in the example, is useful when the representation of the electronics does not comfortably fit into the screen real estate alongside the programming blocks, such as in the case of simulations.

6 DISCUSSION

The goal of this review was to identify common design strategies for representing electronic components in the user interfaces of VPLs for children within the context of physical computing kits. We achieve this goal through the framework used in our analysis, which allowed us to identify What was represented in the interface, How it was represented, and Where it was represented. Our codes reflect common design choices made in the representation of electronics in VPLs for children, and we have discussed how these decisions

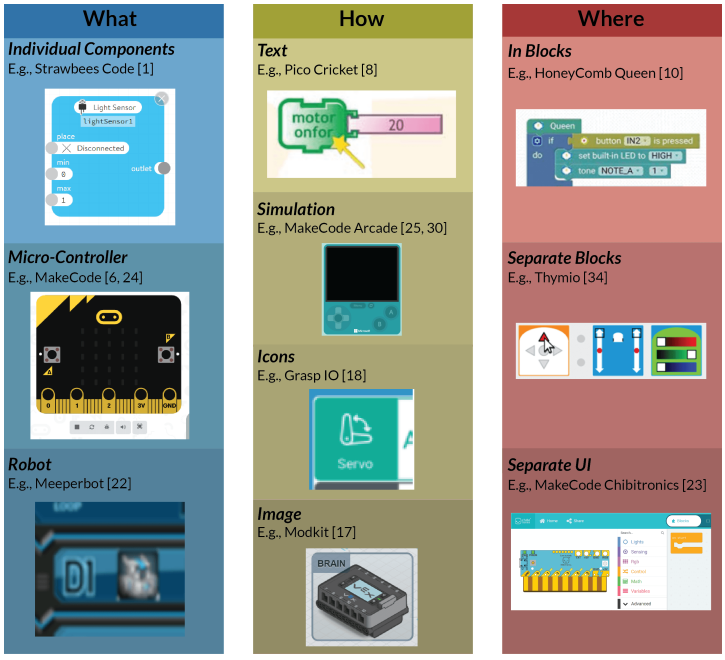

Figure 3: Snippets from Representative VPL interfaces for each code.

What		*How*		*Where*	
Individual Components	46.67%	N/A	35.48%	N/A	40.00%
N/A	36.67%	Text	32.36%	In Blocks	26.67%
Micro-Controller	10.00%	Simulation	16.13%	Separate Blocks	16.67%
Robot	6.67%	Icons	12.90%	Separate UI	16.67%
-	-	Image	3.32%	-	-

Table 1: Our resulting codes, broken down into percentages within each category (each column totals 100%).

impact the scaffolding of the electronics and computation in these interfaces. We now identify future avenues for research and design, based on our findings.

From our results of What is represented in terms of electronics in VPL interfaces, there appears to be a focus on representing smaller electronics components, such as individual pieces of hardware, as opposed to representing larger higher-level units such as assembled robots or micro-controllers. This indicates a preference to scaffold directly between the computation and individual components that it affects, which may support understanding and iterative exploration at a more granular level. We propose that the trend of representing individual electronic components be also supplemented by a larger, more holistic representation of the electronics-enabled physical artifacts, to both provide a low-level and high-level scaffolding of the electronics as they relate to the computation. Future research could investigate the impacts of scaffolding electronics at these varying levels of scale on students' understanding of the relationships between the computation and electronics, and their ability to engage in an iterative tinkering process for creation of computational artifacts.

In terms of How electronics are represented, a surprising majority of cases where electronics were represented in VPLs used text as the means of representation, though there were a number of combined text with icons. Much like Kahn's ToonTalk [20], the use of visual approaches may be more likely to engage the child. However, there has yet to be research on which modality, whether visuals, text, or a combination of both, are more adept at scaffolding understanding between the computation and electronics. Another research direction could be to investigate different design strategies to embed given modalities of the electronics representations in the VPL interface, without overcrowding the screen.

An interesting case in this dimension was the use of simulation, which, for all but one case, was used to represent assembled electronics components such as robots or micro-controllers. One may naturally ask: what is the purpose of a simulation of electronics in a scenario where the physicality of the experience is so integral? In the case of the work of Cabrera et al. [9], whose study of the VPL MakeCode was included in this review, the use of a live simulation impacted the way children interacted with the BBC Microbit micro-controller. Their results showed that in the condition where an interface with a simulation was provided, users spent less time

Physical Computing Kit/VPL	Reference	What	How	Where
Meeperbots	[30, 31]	Robot	Icons	In Blocks
Coji	[54]	N/A	N/A	N/A
Thymio	[40, 43, 48, 49]	Individual Components	Icons	Separate Blocks
Dash	[53]	N/A	N/A	N/A
Blue-Bot	[47]	Robot	Simulation	Separate UI
TALKOO	[21]	Individual Components	Text	Separate Blocks
MakeCode	[9, 34]	Micro-Controller	Simulation	Separate UI
Custom Textile Kit	[12]	Individual Components	Simulation	N/A
Mover Kit	[51]	N/A	N/A	N/A
MakerArcade	[35, 42]	Micro-Controller	Simulation	Separate UI
Pico Cricket	[11]	Individual Components	Text	In Blocks
Strawbees Code	[1]	Individual Components	Icons	In Blocks
MakeCode Chibitronics	[33]	Micro-Controller	Simulation	Separate UI
Cozmo	[23]	N/A	N/A	N/A
Modkit	[25]	Individual Components	Image	Separate UI
Node-RED	[36]	Individual Components	Text	Separate Blocks
Grasp IO	[26]	Individual Components	Icons; Text	In Blocks
Wyliodrin	[37]	Individual Components	Text	In Blocks
MakeCode Mindstorms	[32]	Individual Components	Text	In Blocks
Sphero SPRK+	[44]	N/A	N/A	N/A
Scottie Go!	[16]	N/A	N/A	N/A
Robotis Dream II	[2]	N/A	N/A	N/A
My First Robot	[50]	N/A	N/A	N/A
Photon	[13]	N/A	N/A	N/A
The Scribbler 3 Robot	[19]	Individual Components	Text	Separate Blocks
Roboplus	[15]	Individual Components	Text	Separate Blocks
SunFounder PiCar-V	[46]	Individual Components	Text	In Blocks
UBTECH Jimu Robot MeeBot 2.0	[4]	N/A	N/A	N/A
HoneyComb Queen	[14]	Individual Components	Text	In Blocks
Robo Wunderkind	[5]	N/A	N/A	N/A

Table 2: Our coding breakdown for each of the gathered kits/VPLs.

interacting with the physical device, but exhibited differing interaction patterns. They also found that users who interacted with the VPL that contained a simulation touched the included microcontroller less frequently, but for longer periods of time, likely due to not needing to upload the program to the physical device every time they wanted to test it. Further investigations are needed to fully assess how having a simulation readily available affects behaviors such as tinkering or rapid-prototyping of code.

The results for Where to represent electronics in physical computing kits were particularly revealing. There seems to be a strong preference to place representations of the electronics as close to the computation as possible, either in programming blocks or making up entire blocks themselves. This places the electronics in close proximity to the computation, making a clear connection between the two for students to understand. Additionally, by incorporating the electronics into the blocks, representations of the electronics become directly manipulable by children, perhaps further strengthening their understanding of how changes in the computation reflects in the accompanying electronic components. It would be interesting to investigate how the proximity of electronics representations to the code scaffold understanding of how the two affect each other.

The potential of physical computing kits extends past the learning of computation and fabrication alone. Though not many, there do exist some who use physical computing to also incorporate science learning into the mix [10, 18]. Thus, in the future, as more of these physical computing kits are made specifically for STEM education, there may be a need for similar scaffolding of STEM concepts through the embedding of appropriate STEM-related representations into the VPLs provided to children. Much like how the design strategies that we uncovered support the understanding of electronics, they could also be wielded to support the understanding of concepts in the target subject domain, such as science, where the physical computing activities are being applied.

7 CONCLUSION

We have presented a review of how VPLs designed for children's physical computing kits scaffold understanding between computation and electronics within the kit. We discussed existing ways this scaffolding is achieved, as well as possible directions for future

research and design. This includes providing both low-level and high-level representations of electronics, the increased use of visual representations, and the expanding of these scaffolding design strategies to the science concepts within physical computing kits for science learning. We hope this both aids in the design of future VPLs in physical computing kits for children, paves the way for future avenues of research, as well as broadens the understanding of existing methods VPLs used to scaffold understanding between computation and electronics. One limitation of this work is that the review of physical computing kits conducted was not necessarily systematic and comprehensive in nature.

8 SELECTION AND PARTICIPATION OF CHILDREN

No children participated in this work.

ACKNOWLEDGMENTS

This research was supported by NSF Grant #1934113, *Science Modeling through Physical Computing: Contextualized Computational and Scientific Learning in the Grade 5-6 Classroom.*

REFERENCES

[1] Strawbees AB. 2021. *Strawbees Code.* https://code.strawbees.com/block/
[2] ROBOTIS AMERICA. 2019. *ROBOTIS DREAM How does the program work.* https://www.youtube.com/watch?v=3l_I05xhoGgU
[3] Julia Anghileri. 2006. Scaffolding practices that enhance mathematics learning. *Journal of Mathematics Teacher Education* 9, 1 (2006), 33–52.
[4] Apple. 2021. *UBTECH Jimu Robot MeeBot 2.0 App-Enabled Building and Coding STEM Kit.* https://www.apple.com/shop/
[5] Appysmarts. 2019. *Simple Drawing robot with Robo Wunderkind (STEM DIY project for kids).* https://www.youtube.com/watch?v=NB6Xb_PjbPc
[6] Paulo Blikstein. 2013. Digital fabrication and 'making'in education: The democratization of invention. *FabLabs: Of machines, makers and inventors* 4, 1 (2013), 1–21.
[7] Paulo Blikstein. 2013. Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of the 12th international conference on interaction design and children.* 173–182.
[8] Paulo Blikstein et al. 2015. Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. *Found. Trends Hum. Comput. Interact.* 9, 1 (2015), 1–68.
[9] Lautaro Cabrera, John H Maloney, and David Weintrop. 2019. Programs in the palm of your hand: How live programming shapes children's interactions with physical computing devices. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children.* 227–236.
[10] Sharon Lynn Chu, Genna Angello, Michael Saenz, and Francis Quek. 2017. Fun in Making: Understanding the experience of fun and learning through curriculum-based Making in the elementary school classroom. *Entertainment Computing* 18 (2017), 31–40.
[11] Playful Invention Company. 2021. *Pico Cricket.* https://www.playfulinvention.com/picocricket/index.html
[12] Richard Lee Davis, Chris Proctor, Michelle Friend, and Paulo Blikstein. 2018. Solder and Wire or Needle and Thread: Examining the Effects of Electronic Textile Construction Kits on Girls' Attitudes Towards Computing and Arts. International Society of the Learning Sciences, Inc.[ISLS].
[13] Photon Education. 2017. *Meet Photon - The world's first robot that grows with your child!* https://www.youtube.com/watch?v=nI7ZYbNWFlI
[14] EF ELECFREAKS. 2021. *HoneyComb Queen Kit Electronic Building Blocks.* https://www.amazon.com/HoneyComb-Electronic-Educational-Programming-Programmable/dp/B07QWVW4RK?th=1
[15] Josep Marin Garces. 2018. *Roboplus task bioloid humanoid simple program tutorial.* https://www.youtube.com/watch?v=evAmSv9Qw9g
[16] Scottie Go! 2019. *HOW TO PLAY SCOTTIE GO! - INSTRUCTION.* https://www.youtube.com/watch?v=_hTAmUJttjU
[17] Marcos J Gomez, Marco Moresi, and Luciana Benotti. 2019. Text-based programming in elementary school: a comparative study of programming abilities in children with and without block-based experience. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education.* 402–408.
[18] John Grasel, Wynn Vonnegut, and Zachary Dodds. 2010. Bitwise biology: crossdisciplinary physical computing atop the Arduino. In *2010 AAAI spring symposium series.*
[19] Parallax Inc. 2016. *The Scribbler 3 Robot.* https://www.youtube.com/watch?v=5xCu-Eg3HeE
[20] Ken Kahn. 1996. ToonTalkTM—an animated programming environment for children. *Journal of Visual Languages & Computing* 7, 2 (1996), 197–217.
[21] Eva-Sophie Katterfeldt, David Cuartielles, Daniel Spikol, and Nils Ehrenberg. 2016. Talkoo: A new paradigm for physical computing at school. In *Proceedings of the The 15th International Conference on Interaction Design and Children.* 512–517.
[22] Eva-Sophie Katterfeldt, Nadine Dittert, and Heidi Schelhowe. 2015. Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction* 5 (2015), 3–10.
[23] DIGITAL DREAM LABS. 2021. *Cozmo.* https://www.digitaldreamlabs.com/pages/cozmo
[24] Rong-Hao Liang, Han-Chih Kuo, and Bing-Yu Chen. 2016. GaussRFID: Reinventing physical toys using magnetic RFID development kits. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* 4233–4237.
[25] Modkit LLC. 2021. *Modkit for Vex.* http://www.modkit.com/vex/editor/
[26] Grasp IO Innovations Pvt. Ltd. 2021. *Grasp IO.* https://www.grasp.io/
[27] Sze Yee Lye and Joyce Hwee Ling Koh. 2014. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior* 41 (2014), 51–61.
[28] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 1–15.
[29] Janet Maybin, Neil Mercer, and Barry Stierer. 1992. Scaffolding learning in the classroom. *Thinking voices: The work of the national oracy project* (1992), 186–195.
[30] Meeper. 2021. *Meeper.* https://play.google.com/store/apps/details?id=com.meepertek.meeperbots&hl=en
[31] Meeper. 2021. *MEEPERBOTS.* https://meeperbot.com/pages/meeperbots
[32] Microsoft. 2018. *MakeCode Mindstorms.* https://makecode.mindstorms.com/#
[33] Microsoft. 2021. *MakeCode Chibitronics.* https://makecode.chibitronics.com/#editor
[34] Microsoft. 2021. *micro:bit.* https://makecode.microbit.org/#editor
[35] Microsoft. 2021. *Microsoft MakeCode Arcade.* https://arcade.makecode.com/#editor
[36] Node-RED. 2021. *Node-RED.* https://nodered.org/
[37] Melanie Pinola. 2014. *Wyliodrin Programs the Raspberry Pi with a Drag-and-Drop Interface.* https://lifehacker.com/wyliodrin-programs-the-raspberry-pi-with-a-drag-and-dro-1630107933
[38] Mareen Przybylla and Ralf Romeike. 2014. Physical Computing and Its Scope–Towards a Constructionist Computer Science Curriculum with Physical Computing. *Informatics in Education* 13, 2 (2014), 241–254.
[39] Kanjun Qiu, Leah Buechley, Edward Baafi, and Wendy Dubow. 2013. A curriculum for teaching computer science through computational textiles. In *Proceedings of the 12th international conference on interaction design and children.* 20–27.
[40] Fanny Riedo, Morgane Chevalier, Stéphane Magnenat, and Francesco Mondada. 2013. Thymio II, a robot that grows wiser with children. In *2013 IEEE Workshop on Advanced Robotics and its Social Impacts.* IEEE, 187–193.
[41] Eric Schweikardt and Mark D Gross. 2006. roBlocks: a robotic construction kit for mathematics and science education. In *Proceedings of the 8th international conference on Multimodal interfaces.* 72–75.
[42] Teddy Seyed, Peli de Halleux, Michal Moskal, James Devine, Joe Finney, Steve Hodges, and Thomas Ball. 2019. MakerArcade: Using Gaming and Physical Computing for Playful Making, Learning, and Creativity. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems.* 1–6.
[43] Jiwon Shin, Roland Siegwart, and Stéphane Magnenat. 2014. Visual programming language for Thymio II robot. In *Conference on Interaction Design and Children (IDC'14).* ETH Zürich.
[44] Sphero. 2021. *Sphero SPRK+.* https://www.amazon.com/Sphero-K001RW1-SPRK-App-Enabled-Robot/dp/B01GZ1S7OS?ref_=fsclp_pl_dp_2
[45] Florence R Sullivan and John Heffernan. 2016. Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education* 48, 2 (2016), 105–128.
[46] SunFounder. 2021. *SunFounder PiCar-V Kit V2.0 for Raspberry Pi.* https://www.sunfounder.com/products/smart-video-car?gclid=EAIaIQobChMI6KL-hpP_6gIVgo5bCh26zAmyEAQYCiABEgLg9_D_BwE
[47] Terrapin. 2021. *Blue-Bot.* https://www.terrapinlogo.com/products/robots/blue/blue-bot-family.html
[48] Thymio. 2021. *Program with VPL.* https://www.thymio.org/program/vpl/
[49] Thymio. 2021. *Thymio.* https://www.thymio.org/
[50] Tinkerbots. 2017. *Meet My First Robot.* https://www.youtube.com/watch?v=oBv_odbrVnU
[51] Technology Will Save Us. 2017. *Mover Kit: get kids moving, building & coding.* https://www.kickstarter.com/projects/techwillsaveus/mover-kit-the-first-active-wearable-that-kids-make

[52] David Wood, Jerome S Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of child psychology and psychiatry* 17, 2 (1976), 89–100.
[53] Wonder Workshop. 2019. *Wonder Workshop*. https://www.makewonder.com/
[54] WowWee. 2021. *COJI by WowWee*. https://wowwee.com/coji
[55] Junnan Yu and Ricarose Roque. 2018. A survey of computational kits for young children. In *Proceedings of the 17th ACM conference on interaction design and children*. 289–299.
[56] Junnan Yu and Ricarose Roque. 2019. A review of computational toys and kits for young children. *International Journal of Child-Computer Interaction* 21 (2019), 17–36.