Draw2Code: Low-Cost Tangible Programming for Creating AR Animations

Hyejin Im
hyejin.im@tufts.edu
Tufts University
Medford, MA, USA

Chris Rogers
crogers@tufts.edu
Tufts University
Medford, MA, USA

Figure 1: Draw2Code paper coding block configuration for creating AR animations (left), A physical form of the drawing transformed to a virtual sprite overlaid real-world context through the Draw2Code application on a mobile device (right)

ABSTRACT

Computational thinking is nowadays considered an essential skill in the K-12 educational curriculum. Many tangible computational kits designed for early childhood are either too expensive to benefit a wide range of children or only provide predetermined challenges with limited creative content creation opportunities. In this paper, we investigated low-cost and expressive tangible interfaces that foster computational literacy. We present Draw2Code, a paper-based computational kit for young children to create an interactive AR animation. Children use Draw2Code to make their paper drawing alive as an animated virtual sprite and control it using hand gestures. It exposes children to basic programming concepts through playful and tangible interaction. Results from our initial evaluation with nine child-parent dyads indicate that children ages 5 to 12 successfully used Draw2Code and played with Draw2Code in 33 minutes on average while creating 2 to 5 diverse AR animations based on their ideas. Throughout the session, all children were engaged in computational thinking concepts and practices and learned drawing and gesture-based interactions.

KEYWORDS

Tangible User Interfaces; Computational Thinking; K-12 Education; Augmented Reality

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

ACM ISBN 978-1-4503-8452-0/21/06.
https://doi.org/10.1145/3459990.3465189

ACM Reference Format:
Hyejin Im and Chris Rogers. 2021. Draw2Code: Low-Cost Tangible Programming for Creating AR Animations. In *Interaction Design and Children (IDC '21), June 24–30, 2021, Athens, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3459990.3465189

1 INTRODUCTION

Computational thinking is considered an essential skill for everyone and has been incorporated into the K-12 educational curriculum [7, 30]. Previous research has shown that children from prekindergarten to early elementary school students can learn about fundamental programming concepts such as sequence, loops, parameters [8, 25, 26]. Furthermore, early exposure to STEM activities can benefit as it positively impacts children's self-efficacy towards STEM [10].

The current computational thinking kits can be classified into three groups: physical, virtual, and hybrid [31]. Among these modalities, tangible interaction provides a more intuitive and engaging experience for children with a low level of digital literacy [16, 28]. Examples include Tern, Cubetto, Coding Awbie, and KIBO that use wooden or plastic blocks with external cameras and scanners [1, 2, 17, 27]. On the other hand, Topobo and Google Project Blocks offer more advanced manipulations using custom-built embedded hardware [9, 23], while more expensive. Other kits such as Sheets, Roberto, and Pixel Press [4, 15, 29] use more affordable materials including pen & paper, webcams, and mobile devices. Recent tools leveraged augmented reality to provide a new emerging experience, such as AR Scratch, Code Bits, AR-Maze, Paper Cubes, and CodeCubes [11, 13, 14, 19, 20, 22], although they are not intended for young children ages under nine. The degree of expressing creativity varies by technology. For instance, ScratchJr allows for using custom sprites [12], while Code Bits enable children to organize

predefined paper bits representing different commands [14]. Also, while a broad range of novel interfaces has diverged from screen-based interaction or Direct Manipulation interaction style [18], they have been rarely applied in the coding learning context for young children.

In this work, we present Draw2Code, a paper-based computational kit for young children ages five and older to learn basic programming concepts by creating interactive 2D augmented reality animation. Our work share similarities with the aforementioned technologies but differ in several key dimensions that serve as our design goals: **D1** designing a tangible programming environment that supports natural interaction and physical engagement, **D2** making it low-cost and affordable, **D3** offering creative expression. Draw2Code uses drawing and physical interactions to make the coding activity more participatory and approachable. The output animation is in an AR environment and further controlled by their body gestures, making the engagement visceral and introducing a novel interaction. The use of pen & paper, and widely available mobile devices makes it affordable, while the flexibility of drawing makes the outcome expressive and personally meaningful. In the following sections, we describe the Draw2Code system and a user study.

2 DRAW2CODE SYSTEM

The Draw2Code kit consists of two major components: paper coding blocks and a mobile application. Children can author their own animations by manipulating the paper coding blocks and execute the code with a mobile device. Draw2Code is designed based on the Constructionism theory, providing a tangible programming environment for computational thinking [21]. We further followed the Positive Technological Development (PTD) that proposes six behaviors (e.g., collaboration, content creation, creativity, community building) by allowing users to work together on a shared tablespace and easily incorporate their ideas and environments into AR animations [6]. We also considered the construction kit design principles that emphasize designing for low floors, high ceilings, and wide walls [24], by making the kit easy for novice programmers and encouraging diverse solutions. To make the kit age-appropriate, we minimized text usage, made buttons large, and used straightforward icons. We also paid attention to involving both boys and girls, such as using colors to which both genders can feel a sense of belonging. We used p5.js library [3] and Teachable Machine [5] to recognize and process paper coding blocks as well as body gestures.

2.1 Paper coding blocks

Paper coding blocks consist of the Sprite, Action, and Event blocks with distinguished shapes and colors without text or cumbersome AR markers. The **Sprite** block allows children to design their own character, while the **Action** block controls the character's size and position. The **Event** block enables children to interact with the animation using gestures. Similar to ScratchJr [12], the blocks have connectors between them to provide a constraint to avoid syntax errors while creating a program. Children can build as many coding blocks as they want with accessible materials.

2.2 Coding & Interaction Workflow

Designing a character. Children draw a character on the empty canvas in a **Sprite** block (Fig. 2). If desired, they can *place a small object* on the block to use it as a character. The physical modality makes it easier for them to brainstorm the character's motion trajectories and envision the background scenes that the character would augment.

Coding a behavior of the character. Once the character is drawn, children create an animation using a series of **Action** blocks to define when and how the character animate. They fill the Action blocks and arrange them in a physical space to determine the sequence of animation keyframes (Fig. 2). The number of Action blocks will match the length of the animation. Children can choose one of the **Event** blocks to trigger the animation. Draw2Code currently supports rock paper scissors gestures, while this can be extended to include custom gestures using Teachable Machine [5].

Scanning coding block configurations. Once the program is produced using paper coding blocks, children scan it using a mobile device. When they hover a block, a matching graphic appears, and children tap the Scan button for confirmation (Fig. 3 Left). Children can see the scanned coding blocks on the screen as they scan blocks one by one.

Playing with an interactive AR animation. Children run the program and see the virtual character animating on the screen. They can control the character by a specific hand gesture defined by the chosen **Event** block. It makes programming outcome more engaging for children. They can use any physical background scene by directing a camera toward the background, serving as an additional creative expression dimension for their animated story.

Debugging and iterating. Draw2Code supports debugging by enabling to check the code being executed on the screen while watching the animation (Fig. 3 Right). If necessary, children can change the physical program and scan it. Thus, children can continuously test their code and iterate on the animation sequence design.

3 EVALUATION

We conducted a user study to evaluate the usability of Draw2Code. Our main questions were: **Q1.** Learnability & Usability - How do young children learn and use Draw2Code? **Q2.** Enjoyment & Engagement - What aspects of Draw2Code do children engage most? **Q3.** Usefulness - What programming concepts can children learn using Draw2Code?

3.1 Participants & Procedure

We recruited 13 child-parent dyads through the Tufts Center for Engineering Education and Outreach mailing list. For our analysis, we removed four dyads whose sessions were not successful because of the poor quality of paper blocks, low-performance image recognition of the app in a specific lighting condition, or incomplete post-survey responses. The final nine child participants include 5 females and 4 males (Age M=7.66, SD=2.17, Min=5, Max=12). Each remote study session was conducted at participants' homes for about an hour and was compensated with a $20 gift card. We obtained child assent and parental consent. Parents were asked to

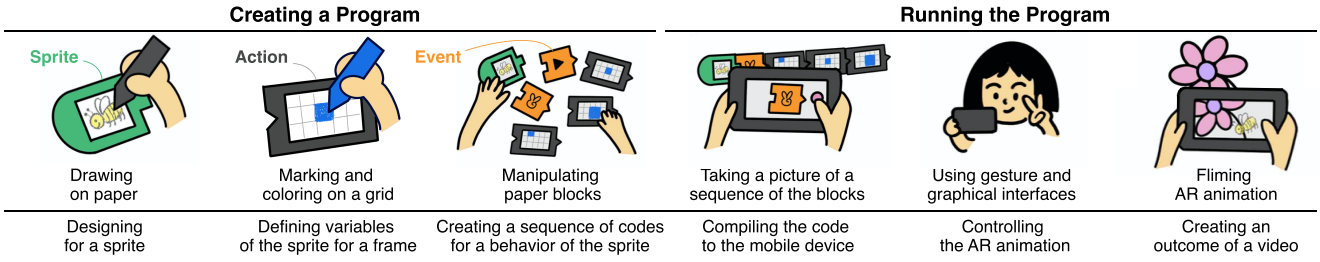

Figure 2: Illustration of Coding & interaction workflow in Draw2Code

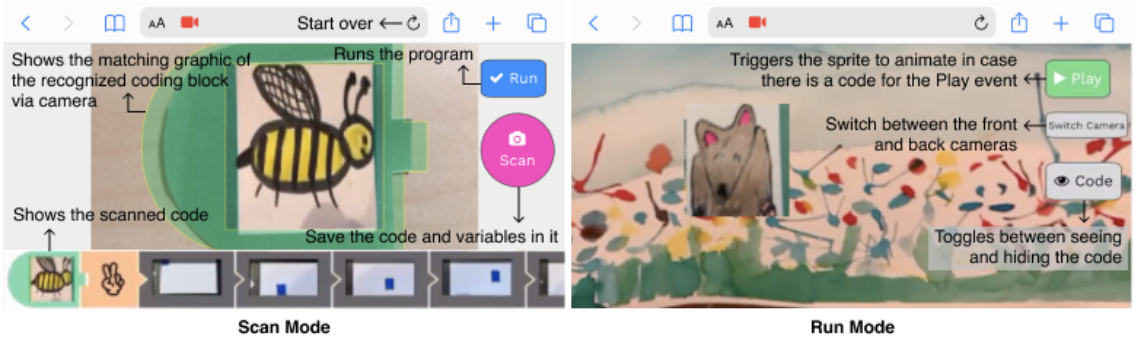

Figure 3: Artifact of the code generated by C9(10) on the Draw2Code interface of the Scan mode (left) and artifact of the AR Animation generated by C12(6) on the Draw2Code interface of the Scan mode (right)

read a provided cheat sheet, print and cut the coding blocks, and prepare drawing pens and an iPhone device in advance. In the study session, we began by asking the child's background including prior coding experience. We then introduced Draw2Code to children with a video tutorial. We asked the children to perform preliminary tasks by identifying and describing each coding block, interpreting a provided Draw2Code program, and identifying syntax errors. In the final task, the children freely explored the Draw2Code system to create their own AR animation. They were free to work with their parents and spend as much time as they want. We concluded the session with a follow-up interview and post-survey with both children and parents.

3.2 Results

Learnability & Usability. All children successfully used Draw2Code without direct adult assistance once they figured out how to use it. 5-6 years old children struggled to explain blocks and program verbally, but they could complete the programming task (Fig. 4 Left). Several parents mentioned that its simplicity and intuitiveness helped children as young as five or with no coding experience understood easily. *"She figured out how to do it really quickly"*-P1(5), *"She was all on her own and didn't need my help at all."*-P9(10). The length of code produced was 5 to 10 blocks. Two children wanted to use more action blocks, and their parents printed them out right away. While two children said it was good as it was, a few usability issues were identified, reflecting the post-survey results (Fig. 5 Left). All dyads wanted to utilize a graphical interface to modify their code on the screen instead of starting over scanning blocks. 5 of 9 dyads mentioned that they struggled with scanning blocks (Fig. 6b), "Lining up correctly on an image was a little hard for me."-C9(10). One child, C11(7), had to scan the same code seven times to capture the image completely. While most children successfully used gesture commands, a few 5-6 years old children were not sure how to use gesture commands at first, asking *"Where should I show my hands?"*-C1(5). Some children ages 7 and older suggested more gestures such as thumbs up and clap. We also observed collaborative behaviors between child and parent. Some parents brought an object that children could use as background. One dyad worked in parallel on different tasks, *"While you are drawing a forest [background], I will complete your code [coloring]"*-P12(6).

Enjoyment & Engagement. The average free play length was 32.55 minutes (*SD*=8.23, Min=24, Max=48), indicating high engagement (Fig. 4 Right). Five children wanted to play more, but they were forced to end the session due to the time constraint. One parent said *"I think she could do this thing for hours"*-P6(10). Parents said while the programming concepts were similar as seen in other coding platforms such as Scratch, Draw2Code was *"More engaging, more movement, more creativity"*-P12(6) and suitable for beginners. They pointed out several aspects that made Draw2Code engaging. First, all children liked drawing and watching their drawings on the screen regardless of gender or age. *"It was really fun and cool to watch my bee across the screen!"*-C9(10). *"Incorporating drawing onto the screen gave my child an opportunity to personalize work and led to the desire to explore the app further"*-P2(12). Second, they appreciated the newly introduced drawing and gesture-based interactions to program and control, *"I like how it's different. Normally you code where you tell it to do this [predefined command] but marking it on*

	Tasks	Success Rate	Example Quotes for Description
Preliminary tasks after tutorial	Identify the Sprite block Describe the Sprite block	100% (9/9) 100% (9/9)	"That is going to be your little symbol that moves around the screen."-P7(8)
	Identify the Event block Describe the Event block	100% (9/9) 78% (7/9)	"You show the peace sign then that's when your character would show up."-C11(7)
	Identify the Action block Describe the Action block	100% (9/9) 78% (7/9)	"You can color that in a certain space and your character will be mapped in that space."-C6(7)
	Interprete a program	67% (6/9)	"It would appear when you show the peace sign and it will start out small and get bigger."-C7(8)
	Identify syntax errors	100% (9/9)	"There's a missing block because those blocks don't go together."-C12(6)
Free play	Program for AR animation	100% (9/9)	"Whale swimming like a spiral motion."-C1(5)

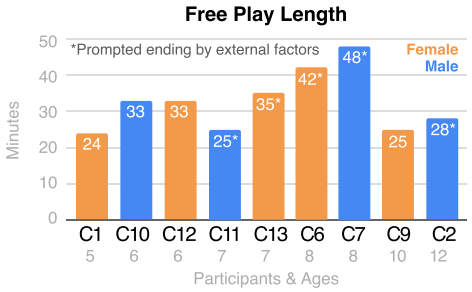

Figure 4: Results of the Task Success Rate (Left) and Free Play Length (Right)

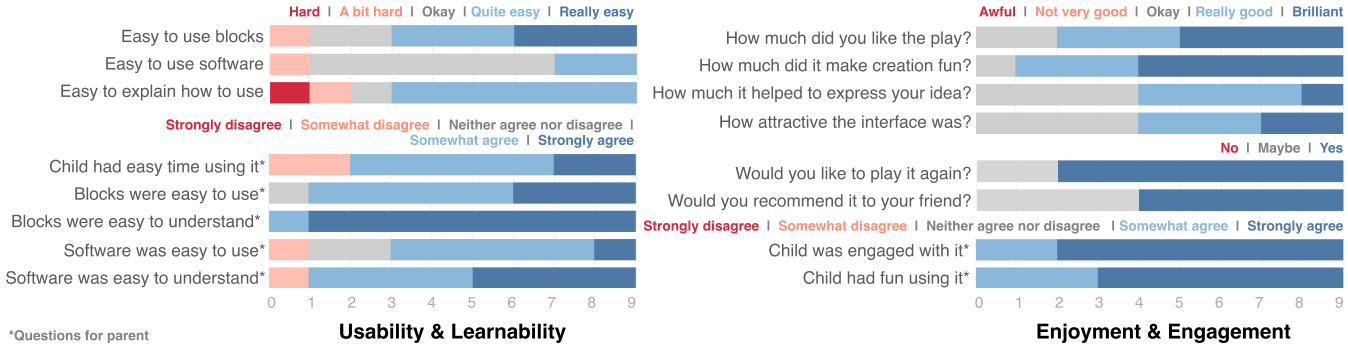

Figure 5: Results of the post-survey: Usability & Learnability questionnaire (left) and Enjoyment & Engagement questionnaire (right)

a grid, I never know you could do that. It was much more helpful."-C9(10) and *"In other coding classes, mostly, with your hands on your keyboard or the mouse. [With Draw2Code] Draw or use your hands [gesture]."*-C1(5). Finally, being able to combine physical and virtual worlds via AR resulted in the diversity of creations (Fig. 6c-f), *"You can put it up into the background or selfie. Really fun."*-C9(10). As evidence of it, each child created 2 to 5 different animations (M=3.22, SD=0.83) based on their ideas and some children actively walked around their rooms to find proper places to play their AR animations. Children used real-world elements as a background (e.g., actual view/paintings/photos/drawing) *"Little me jumping on the snow."*-C11(7) (Fig 6c and Fig 3 Right), another character (e.g., physical objects/real people) *"Creepy monster is chasing me!"*-C6(8) (Fig 6d-f), and sound, *"[a bee coming to her face] Ouch!"*-C9(10). However, two children expressed a slight disappointment at their creations because of either lack of their drawing skills or too simple animation effect, *"It doesn't move like a whale."*-C1(5).

Usefulness for computational thinking. Parents reported that children were engaged in computational thinking such as decomposition by breaking down their design ideas and steps, abstraction by transforming character's appearance to the variables of size and position, and algorithms by creating a specific sequence of coding blocks in order to make an animation. Although the current Draw2Code did not support advanced programming functions, children ages 7 and older came up with the idea of loop and parallel programming by scanning the same piece of code repeatedly rather than creating a long code and creating a code using multiple sprites that move at once (Fig. 6a). Most children debugged watching their creations using the interface and iterated their design. They also reused existing code for new projects. We also found that children tried to be resilient and optimistic when they encountered challenges. They troubleshooted and tried over and over again until they got it to work. Several children ages 7 and older were curious about the technologies used in Draw2Code, such as AR, computer vision, and AI, which were new to them. One asked about how AR worked, *"How do you put my drawing in the physical world?"*-C7(8). A few dyads had a conversation about why the app had a hard time differentiating blocks and figured out reasons: *"I see the problem. It is because my table has the same color as the other block"*-C9(10) and *"It confuses my hand and my face. It just gets as hand if it sees my skin"*-P6(8). Children were also engaged in learning art and filming to make their creations look more visually pleasing.

4 DISCUSSION AND CONCLUSION

The user study revealed opportunities for improvements of Draw2Code and provided insights on future directions.

Our study revealed that scanning was the most challenging part, especially for children who had weaker fine motor skills since it requires lining up an image completely. Enabling automatic image extraction for scanning blocks would address this issue by making scanning easier. Also, Our study identified an issue that children had to scan similar code repeatedly if they wanted to change a part of the code. Enabling code modification on the screen would reduce workload and encourage code iterations. We found that

Figure 6: Images of children using Draw2Code (a, b) and screenshots of AR animations generated by children (d-f)

children with limited drawing skills might not be satisfied with their programming outcomes, and drawing supports such as stickers and templates would be helpful for them.

Our study indicated that a paper-based programming interface using computer vision naturally encouraged collaboration by allowing a child and a parent to work on a shared tablespace. Also, involving varying physical activities (e.g., drawing, coloring, scanning, filming) allowed them to work in parallel, easing collaboration with multiple people. Our study involved child-parent dyads, but a large-scale user study can be an interesting future study.

Our study showed that children could begin learning about advanced technologies behind a computational kit. While having a limited number of coding blocks and simple steps made Draw2Code accessible to young novice programmers, children ages 7 and older wanted to know how AR, computer vision, and AI worked in Draw2Code. Computational kits can scaffold up to introduce advanced computing technologies and help with increasing their technological fluency.

In this work, we designed and implemented Draw2Code, a low-cost and expressive tangible interface for computational thinking. We conducted a user study with nine child-parent dyads to evaluate the usability of Draw2Code. For future work, we plan to improve the system by addressing the issues identified in the user study.

ACKNOWLEDGMENTS

Thanks to Nam Wook Kim and Khushbu Kshirsagar for your valuable feedback. We also thank colleagues, children, and parents at Tufts Center for Engineering Education and Outreach for their support and help.

REFERENCES

[1] [n.d.]. Coding Awbie. https://www.playosmo.com/en/coding/. Accessed: 2021-05-10.
[2] [n.d.]. Cubetto. www.primotoys.com. Accessed: 2021-05-10.
[3] [n.d.]. p5.js. https://p5js.org/. Accessed: 2021-05-10.
[4] [n.d.]. Pixel Press Floors. http://www.projectpixelpress.com/floors. Accessed: 2021-05-10.
[5] [n.d.]. Teachable Machine. https://teachablemachine.withgoogle.com/. Accessed: 2021-05-10.
[6] Marina Bers, Alicia Doyle-Lynch, and Clement Chau. 2012. Positive technological development: The multifaceted nature of youth technology use towards improving self and society. *Constructing the self in a digital world* (2012), 110–136. https://doi.org/10.1017/CBO9781139027656.007
[7] Marina Umaschi Bers. 2008. *Blocks to robots: Learning with technology in the early childhood classroom.* Teachers College Press New York, NY.
[8] Marina Umaschi Bers, Louise Flannery, Elizabeth R Kazakoff, and Amanda Sullivan. 2014. Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education* 72 (2014), 145–157. https://doi.org/10.1016/j.compedu.2013.10.020
[9] Paulo Blikstein, Arnan Sipitakiat, Jayme Goldstein, João Wilbert, Maggie Johnson, Steve Vranakis, Zebedee Pedersen, and Will Carey. 2016. Project Bloks: designing a development platform for tangible programming for children. *Position paper, retrieved online on* (2016), 06–30.
[10] Yu-Hui Ching, Yu-Chang Hsu, and Sally Baldwin. 2018. Developing computational thinking with educational technologies for young learners. *TechTrends* 62, 6 (2018), 563–573. https://doi.org/10.1007/s11528-018-0292-7
[11] Barbara Cleto, João Martinho Moura, Luís Ferreira, and Cristina Sylla. 2018. CodeCubes-Playing with Cubes and Learning to Code. In *Interactivity, Game Creation, Design, Learning, and Innovation.* Springer, 538–543. https://doi.org/10.1007/978-3-030-06134-0_58
[12] Louise P. Flannery, Brian Silverman, Elizabeth R. Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: Support for Early Childhood Learning through Computer Programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (New York, New York, USA) *(IDC '13).* Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2485760.2485785
[13] Anna Fusté, Judith Amores, David Ha, Jonas Jongejan, and Amit Pitaru. 2017. Paper cubes: evolving 3D characters in augmented reality using recurrent neural networks. In *31st Conference on Neural Information Processing Systems (NIPS 2017).* 31–33.
[14] Sidhant Goyal, Rohan S. Vijay, Charu Monga, and Pratul Kalita. 2016. Code Bits: An Inexpensive Tangible Computational Thinking Toolkit For K-12 Curriculum. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction* (Eindhoven, Netherlands) *(TEI '16).* Association for Computing Machinery, New York, NY, USA, 441–447. https://doi.org/10.1145/2839462.2856541
[15] Michael S. Horn, Sarah AlSulaiman, and Jaime Koh. 2013. Translating Roberto to Omar: Computational Literacy, Stickerbooks, and Cultural Forms. In *Proceedings of the 12th International Conference on Interaction Design and Children* (New York, New York, USA) *(IDC '13).* Association for Computing Machinery, New York, NY, USA, 120–127. https://doi.org/10.1145/2485760.2485773
[16] Michael S. Horn, R. Jordan Crouser, and Marina U. Bers. 2012. Tangible Interaction and Learning: The Case for a Hybrid Approach. *Personal Ubiquitous Comput.* 16, 4 (April 2012), 379–389. https://doi.org/10.1007/s00779-011-0404-2
[17] Michael S. Horn and Robert J. K. Jacob. 2007. Tangible Programming in the Classroom with Tern. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems* (San Jose, CA, USA) *(CHI EA '07).* Association for Computing Machinery, New York, NY, USA, 1965–1970. https://doi.org/10.1145/1240866.1240933
[18] Robert J. K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. 2007. Reality-Based Interaction: Unifying the New Generation of Interaction Styles. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems* (San Jose, CA, USA) *(CHI EA '07).* Association for Computing Machinery, New York, NY, USA, 2465–2470. https://doi.org/10.1145/1240866.1241025
[19] Qiao Jin, Danli Wang, Xiaozhou Deng, Nan Zheng, and Steve Chiu. 2018. ARMaze: A Tangible Programming Tool for Children Based on AR Technology. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (Trondheim, Norway) *(IDC '18).* Association for Computing Machinery, New York, NY, USA, 611–616. https://doi.org/10.1145/3202185.3210784

[20] Chien-Yu Lin and Yu-Ming Chang. 2015. Interactive augmented reality using Scratch 2.0 to improve physical activities for children with developmental disabilities. *Research in developmental disabilities* 37 (2015), 1–8. https://doi.org/10.1016/j.ridd.2014.10.016

[21] Seymour Papert and Idit Harel. 1991. Situating Constructionism. In *Constructionism*, Seymour Papert and Idit Harel (Eds.). Ablex Publishing Corporation, Norwood, NJ, Chapter 1. http://www.papert.org/articles/SituatingConstructionism.html

[22] Iulian Radu and Blair MacIntyre. 2009. Augmented-Reality Scratch: A Children's Authoring Environment for Augmented-Reality Experiences. In *Proceedings of the 8th International Conference on Interaction Design and Children* (Como, Italy) *(IDC '09)*. Association for Computing Machinery, New York, NY, USA, 210–213. https://doi.org/10.1145/1551788.1551831

[23] Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. 2004. Topobo: A Constructive Assembly System with Kinetic Memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 647–654. https://doi.org/10.1145/985692.985774

[24] Mitchel Resnick and Brian Silverman. 2005. Some Reflections on Designing Construction Kits for Kids. In *Proceedings of the 2005 Conference on Interaction Design and Children* (Boulder, Colorado) *(IDC '05)*. Association for Computing Machinery, New York, NY, USA, 117–122. https://doi.org/10.1145/1109540.1109556

[25] Amanda Strawhacker and Marina Umaschi Bers. 2019. What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development* 67, 3 (2019), 541–575. https://doi.org/10.1007/s11423-018-9622

[26] Amanda Sullivan and Marina Umaschi Bers. 2016. Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education* 26, 1 (2016), 3–20. https://doi.org/10.1007/s10798-015-9304-5

[27] Amanda A Sullivan, Marina Umaschi Bers, and Claudia Mihm. 2017. Imagining, playing, and coding with KIBO: using robotics to foster computational thinking in young children. *Siu-cheung KONG The Education University of Hong Kong, Hong Kong* 110 (2017).

[28] Cristina Sylla, Pedro Branco, Clara Coutinho, and Eduarda Coquet. 2012. TUIs vs. GUIs: comparing the learning potential with preschoolers. *Personal and Ubiquitous Computing* 16, 4 (2012), 421–432. https://doi.org/10.1007/s00779-011-0407-z

[29] Kazuki Tada and Jiro Tanaka. 2015. Tangible programming environment using paper cards as command objects. *Procedia Manufacturing* 3 (2015), 5482–5489. https://doi.org/10.1016/j.promfg.2015.07.693

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

[31] 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. https://doi.org/10.1016/j.ijcci.2019.04.001