FIRST GRADERS COORDINATION OF COUNTING AND MOVEMENTS ON A GRID WHEN PROGRAMMING WITH TANGIBLE BLOCKS

Abigail Erskine
Purdue University
aerskin@purdue.edu

Laura Bofferding
Purdue University
lbofferd@purdue.edu

Sezai Kocabos
Purdue University
skocabos@purdue.edu

Haoran Tang
Purdue University
tang474@purdue.edu

As elementary students begin to program using tangible blocks, they must coordinate their use of counting with the movements, directions, and numbers they use to move a character. In our study, we analyzed 13 first graders' first attempts at coordinating these elements when playing a programming game on the iPad that used tangible programming blocks. We further analyzed how their programs changed over the six sessions. Our results highlight the challenges students faced when counting on a grid, representing movements with numbers, and distinguishing between movement blocks. We also present factors that influenced their improvements. The results indicate that game hints supported some students' use of numbers, while the highlighted path helped some and challenged others. Partner talk and having the opportunity to make iterative changes in their code also supported some groups.

Keywords: Problem Solving, Computational Thinking, Computing and Coding, Number Concepts and Operations

Research in K-2 settings has shown a correlation between programming and mathematics scores (e.g., Grover, et al., 2016; Lewis & Shah, 2012; Kocabas et al., 2021). Students as young as three years old can learn to program in visual or tangible alternative programming environments such as Creative Hybrid environment for computer Programming (CHERP, Bers, 2010; Bers et al., 2014; Elkin et al., 2016; Sullivan & Bers, 2016), which pairs programming with a set of physical blocks. These and similar alternative environments reduce the chances of a program not being able to execute (Berland et al., 2013) and support young students by offering visual feedback and tangible manipulatives (Brusilovsky et al., 1997; Mladenovi et al., 2016). They also typically incorporate a physical or digital character (e.g., a turtle or a robot) and programming blocks to give the characters instructions. Blocks may include movements (e.g., walk, turn, jump), directional arrows, and numbers to indicate how many movements to execute (McNerney, 2004, see also Bofferding et al., 2022). One study found that following a one-week CHERP robotics and programming intervention, preschoolers and kindergarteners were more likely to have higher sequencing scores (Kazakoff et al., 2013). Likewise, first and third graders' performance in fixing mathematics bugs of double counting in the pretest were highly correlated with fixing programming bugs of double counting (Kocabas et al., 2021), indicating that learning programming can enrich mathematics content knowledge and problem-solving skills (Fessakis et al., 2013; Friend et al., 2018; Lewis & Shah, 2012). In one study, kindergarteners solved a series of programming puzzles to lead a ladybug to a leaf to hide herself, which also supported their counting and number comparison (Fessakis et al., 2013).

However, prior research has shown that young students struggle with counting in programming (e.g., Bofferding et al., 2020; Kocabas et al., 2019, 2021) and mathematics (e.g., Battista, 1999, 2010; Battista et al., 1998; Fuson, 2012). One counting issue younger students experience is double counting the same space or object twice. For example, three to five year olds made more double counting errors when objects were shown in a disorganized manner than when they were organized (Fuson, 2012; see also Kocabas et al., 2021).

Double counting often arises in structures with horizontal and vertical dimensions, such as arrays. Students have to monitor their count while coordinating their horizontal and vertical position (Risley et al., 2016). Children are good at keeping track of one thing at a time but may have more difficulty coordinating two aspects. For example, second graders who do not demonstrate columns and row structures may double count when columns and rows overlap (Battista, 1999, 2010; Battista et al., 1998). Likewise, when programming, first and third graders double counted the spaces when a column and a row overlapped on a programming path (Kocabas et al., 2019, 2021). Other counting issues may arise due to different movement pieces involved in programming (e.g., movements that move one space versus multiple spaces). The goal of this work was to explore early elementary students' counting related to space and movements in a tangible programming environment to illustrate how they navigate these issues and what helps them in this process. Therefore, we explored the following research questions:

1. When and how do first graders count the spaces they need Awbie to move in one or more directions?
2. How do they coordinate their counting of spaces and use of number blocks?
3. How do they interpret and coordinate differences in walking versus jumping with their counting and use of numbers?
4. What challenges do they face with counting and their use of programming blocks and how do they overcome them (if at all)?

Methods

Participants and Setting

We conducted this study in a midwestern elementary school in the United States where 11% of students were designated as English language learners, and 45% of students were eligible for free and reduced meals. The data in this study comes from a larger study focusing on 29 first and 28 third graders' commenting and debugging practices in programming and mathematics. Students participated in a pre-test, six 20-minute playing sessions, and a post-test in the larger study. For this analysis, we focus on 13 students from two of the first-grade classes (two of them left at different points in the study, so part of the analysis focuses on the remaining 11) as they played the game. During the sessions, each grade-level pair (and one student working alone) met with a researcher at a table in the hallway and worked to advance through different levels of the Coding Awbie game. Each session was video recorded, and researchers also took notes so we could analyze their choices in using the coding blocks to move the character (Awbie).

Materials

The programming game students used, Coding Awbie, uses a series of movement blocks with direction arrows and number blocks that attach to the movement blocks to control how many times Awbie does that movement for each block. The movement blocks include a walk, jump, or grab, and the directions include up, down, left and right. The number blocks range from 1 to 5. Students can combine one movement block and a number or can stack several movement blocks with numbers to make longer programs (a movement block without a number defaults to a movement of 1). Students play the game by arranging their blocks in front of the iPad. A mirror attached to the iPad captures the code and shows the potential path represented by the code highlighted on the iPad screen. When the student presses a tangible play button, the character moves on the screen. If a student blocks the mirror while playing, the iPad may misinterpret the code. Figure 1 shows an example program with three lines of code.

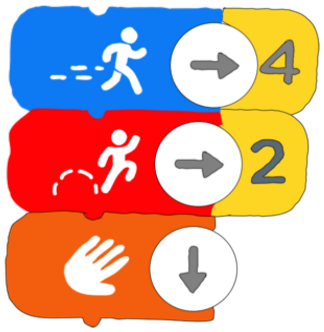

Figure 1: Example of Programming Code

The first line of code in Figure 1 shows *Walk Right 4*, which would move Awbie four spaces to the right from his current spot. The second line of code tells Awbie to *Jump Right 2* times. The jump command enables Awbie to skip a square, so Awbie would jump to the second square and then land on the fourth square from where he started. Awbie can jump over small items like bushes. Students have the ability to turn the arrow to change the direction, which we see in the third line of code where the arrow points down. The grab block keeps Awbie in his current space, but he grabs items (e.g., strawberries) in the square next to him in the given direction.

All groups started on the first level of the game, which we called Forest 1 (see Figure 2 for a map of the first level and screenshot of the starting point of each level). If they finished a level before their time was up, they continued on to the next level. They restarted any level they did not finish during their next session. The researchers encouraged pairs to work together and take turns using the coding pieces or running the program. Each of the six sessions lasted 20 minutes. For three of the six sessions, students spent part of the time studying and explaining worked examples of programs (5 to 8 min). We gave pseudonyms to students, so the pairs included Duck1 & Duck2, Duck3 & Duck4, Duck5 & Duck6, Bat1 & Bat2, Bat6 & Bat7, and Bat8. Bat2 left after the third session, at which point Bat8 played with Bat1. Bat7 left after session four, at which point Bat6 played alone.

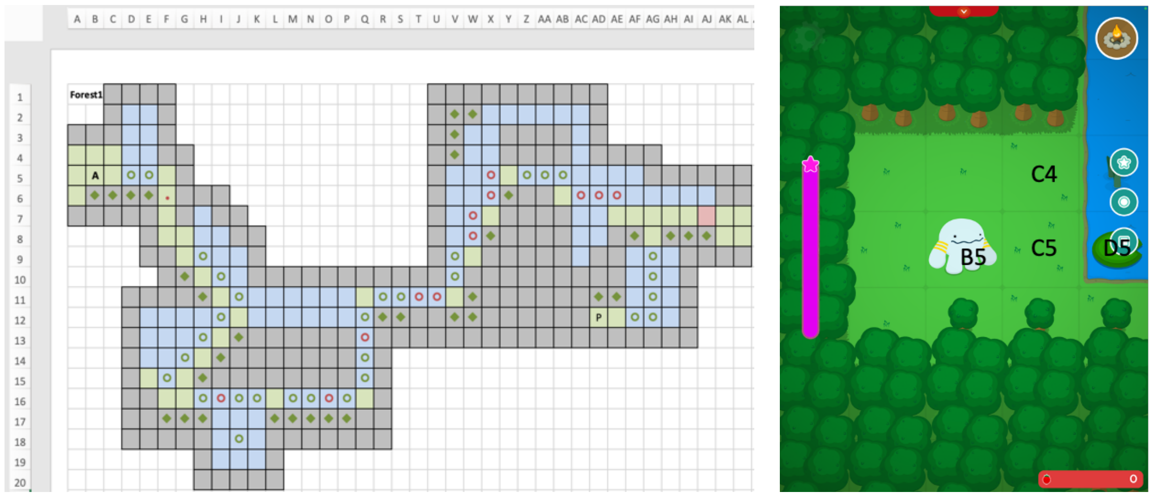

Figure 2: Forest 1 Map and Screenshot of Beginning Position
Description: A = location where Awbie starts. Green space = Grass. Blue space = Water. Red = Strawberry. Circle = Lilypad. Dark green = Tree. P = Pie. Red space = End of level.

Data Analysis

We were interested in how the student pairs played and navigated their counting and coordination of the movements. We coded for instances where students double counted one of the spaces when counting: the starting square (i.e., they included the square where Awbie was standing in their count of spaces to move) or a corner when changing directions (i.e., they counted the square as an ending point for one direction and a beginning point for the new direction). We also tried to make sense of other uses of numbers, such as whether they used numbers as labels for spaces to move to, using numbers as labels for the line of code, or using numbers in patterns or randomly. Finally, we also analyzed how they made sense of the distance they wanted Awbie to move and whether they used numbers that were too high or low, whether they broke up distances with two or more numbers, and whether and how they adjusted their counts.

Students' use of numbers could also vary depending on their interpretation of how the different movement pieces worked. For example, if they thought the jump moved one space (like a walk), their use of numbers would be different than if they knew the jump skipped over a space. Therefore, we also identified times when students were using the jump or grab blocks as if they moved just one space. For both counting and movement blocks, we also made notes for when their use changed and what factors might have influenced their change (e.g., the researcher said something, the game gave them a hint, etc.).

Findings

Student Pairs' Attempts to Use Numbers and Movements on Session 1

Bat6 & Bat7 were the only ones who started off playing (did not do worked examples) and were also the only ones who scrolled ahead to see where the path went before running their first line of code. However, they initially organized their coding pieces from the bottom to the top and used a mixture of jump and walk commands with random numbers. Bat 7 said, "Awbie cannot jump over trees" when their first line of code (Jump Up 5) for their initial program caused Awbie to bounce off the trees. They may have misinterpreted the jump as a walk and kept using it even when the hint showed up as Walk Right 1. They switched to only using the number 1 on their movements when getting the hint, but they often paid more attention to how many movements to make than on where they were going or stopping.

The other six groups initially only saw up to space D5 initially on the screen because they did not scroll to see where the path went; however, they had all explored some worked examples involving the coding pieces. Duck5 and Duck6 successfully moved Awbie to D5 (a lilypad on the water), but because he stopped there, Awbie fell in the water and went back to C5. Once they saw more of the path, they correctly used numbers and navigated the columns and rows to have Awbie Walk Right 3 to F5 and Walk Down 3 to F8. They were going to put a Jump Right 1 next but saw that the highlighted path showed his potential movement would land him in the water, so they changed it to Walk Right 1, ending at G8. At this point, one of the girls noticed that the path looked like stairs, and they were able to successfully program Awbie to Walk Down 1, Walk Right 1, multiple times in a row. They fell in the water a couple times because they did not move down enough times and ended on a lilypad, and they did not try the jump anymore that session.

The second group, Duck3 and Duck4 first made programs separately. Duck3 wanted to Walk Right 1 two times to get to the lilypad on D5 and then Jump Right 1 so the screen would move. Instead, they played Duck4's code, which involved a series of walks and jumps and got them to F5 (hitting a few trees that they had not seen). From F5, they played Walk Down 3, correctly counting to get to F8. Third, Bat8 kept the Jump Right 3 he had been using on a worked example before playing and used it again without knowing where it would go. He kept running previous lines of code multiple times until the researcher would remind him to change the code. For example, when he got to the part that looked like stairs, he correctly played Walk Right 1, Walk Down 1 and ran this code repeatedly, even after falling in the water several times at K12. Although he correctly counted spaces, he then added new movements onto part of his old code, so he often moved too far.

Fourth, and similar to Bat8, Bat1 and Bat2 also reused parts of their previous code. After initially jumping right and then left, Bat1 and Bat2 ended up one space away from their starting square. However, they quickly adjusted their code to successfully Walk Right 3 the correct number of spaces to get to F5. They kept the Jump Right 3 from their first line of code, which made them bounce off the trees, but then they correctly programmed Awbie to Walk Down 3. They often added or changed one correct movement but continued to keep some commands from previous lines of code and used numbers randomly, which hampered their progress. Fifth, Bat3 and Bat5's initial difficulty corresponded to them using the jump and walk blocks as if they both moved one space (potentially because they could not see where they were going). When that did not work for them, they tried just using jumps and added on numbers in sequence with random directions (i.e., Jump Right 1, Jump Right 2, Jump Up 3). They continued using sequences of numbers and a mix of walk and jump blocks.

Finally, Duck1 & Duck2 started off using numbers as locations for where they wanted Awbie to move to. For example, their first three lines of code were Walk Right 1, Walk Right 2, Walk Right 4, but they only needed the third line. Instead, because they had already moved right three, their third line of code made Awbie hit a tree and bounce back. Their continued number use appeared random, and Awbie often fell in the water before completing a step. Some of their difficulty may have been due to forgetting to turn the directional arrows.

Student Pairs' Progress Across Sessions 2-6

Bat6 and Bat7. During sessions 2 and 3, Bat6 and Bat7 continued to build their programs from the bottom up (instead of top down) and sometimes scrolled to a future part of the path when building their code. Although they saw the structure in rows and columns, Bat6 thought the Jump block moved one space, plus often moved one space short. On the other hand, when Awbie was at G9, Bat7 thought Walk Right 3 would move Awbie diagonally along the lily pads. Bat6 tried to show her that it would make Awbie walk horizontally into the water, although she did not listen. They continued with these difficulties, reversing their code, even after following the program's hint showing them the correct way to combine pieces. After they examined worked examples at the beginning of session 4, they started placing their coding pieces in the correct order. Bat6 still thought the Jump would move one space until she played alone during session 5 (Bat7 moved) and noticed feedback from the game on where the jump would move Awbie. By this time, she also correctly counted the spaces to span longer distances.

Bat1 and Bat2, Bat8. Bat1 & Bat2 did not make much progress in sessions 2 and 3. From B5, they started off using Walk Right 1, Walk Right 1, Walk right 1, which did not get them far enough to F5. They kept trying to use a combination of jumps and walks with high numbers, and even ignored or did not follow the hint when it came up in both sessions. During these same sessions, Bat8 once again reused one set of codes until he got a hint from the game (Walk Right 1, Walk Down 1) at which point he repeated that code. At G16, when that code did not work, he changed the second line to Walk Right 1. Finally, he decided to change a number and did Walk Right 5 but also kept Walk Right 1, moving Awbie too far right and landing him in the water. He continued using this code (and going too far), sometimes changing the directions incorrectly or changing the code when he got a new hint. He repeated this same process in session 3, only he started using more jump blocks instead of walk blocks.

Starting at session 4, Bat1 and Bat8 played together because Bat2 moved. Bat1 primarily took charge of the pieces with input from Bat8. In session 4, after they got a hint to Walk Right 1, Walk Down 1, they often kept these pieces and just changed their numbers or directions. Therefore, even when they got to the point where they needed to Walk Up 5, they programmed Walk Up 5, Walk Up 5, once again using both pieces. They continued to change their code in reaction to Awbie not doing what they wanted after trying to reuse the code they already had, and on one occasion they double counted the starting square. By session 5, they were using the movements and numbers more intentionally, and had fewer instances of landing in the water. However, they still had an instance of double counting the initial square even in session 6. In both cases, the double counting happened on the first line of code for a new level.

Bat3 and Bat5. During sessions 2 and 3, Bat3 and Bat5 continued to use random movements, directions, and numbers without counting the spaces, although, Bat3 demonstrated some insight into how they could use a walk movement to get a strawberry they had jumped over. They stopped using numbers other than one after they got a hint to Walk Right 1, Walk Down 1 in session 3. In session 4, they used the movements more intentionally and started counting the number of spaces they needed to move. However, when they were at G16 and wanted to span the large distance to Q16 and then up to Q11, they ran out of Walk blocks. The researcher suggested they use numbers. They did this, plus also used Grab blocks, which they thought would move one space. They saw the structure in the columns and rows without any double counting, although they did think they only needed to move Awbie up four spaces instead of five from Q16 to Q11.

By session 5, they were more intentional and accurate in their use of numbers and directions. They started using the highlighted path, which helped them fix a Walk Down 3 (that was going to make Awbie run into a tree) to Walk Down 2. They also used addition facts for larger distances in sessions 5 and 6. For instance, at G15, Bat3 mentioned that they needed to Walk Down 1 (pointing to the screen) and sideways 4." Bat5 then changed the Walk Right 4 to Walk Right 5, considering the extra space they needed to move. While looking further on the path, they saw the additional spaces and added on another Walk Right 4 and Walk Right 1.

Sometimes, because they would alter previous lines of code without removing extra ones, their use of the highlighted path misled them into using numbers incorrectly. For example, at Q16, they had just finished running Walk Down 1, Walk Right 5, Walk Right 4, Walk Right 1. Since they need to go up, they turned the arrow on the first command up and counted the four lilypads. After moving the four from the third line of code up, they noticed the path did not go high enough and swapped it out for the five from the second command, giving them Walk Up 5, Walk Right 1, Walk Right 1, Walk Right 1. They then counted the four lilypads from R11 to U11 and put a four on their second line, then quickly realized it would need to be a five, giving the Walk Up 5, Walk Right 5, Walk Right 1, Walk Right 1. Because they still had the two extra Walk Right 1's on the end, the path showed that Awbie would walk seven spaces right and hit a bush. They used trial and error to swap out numbers until they found that Walk Right 3 would work. Because they did not realize the last two Walk Right 1's were showing up in the path, they ended up changing those so Awbie did not move as they expected.

Duck1 & Duck2. Although Duck1 and Duck2 appeared to be using numbers randomly in session 1, they started session 2 by considering how many spaces they wanted to move and used one block per space. Sometimes, they did not accurately account for how much space they needed to move. For example, when Awbie was at C5, they programmed him to Walk Right 1, Walk Right 1, Walk Down 1. This led Awbie to stop on a lilypad and fall into the water. Correcting their count the second time, they added in another Walk Right 1 block so that he would walk right three times before walking down. Later, when one partner was going to add an extra Walk Down 1, his partner corrected him, "I already did the down." Another problem they encountered was thinking the jump would move one space. During session 3, when Awbie made it to I13, they once again confused the jump with a walk and also double counted the initial square, thinking Jump Left 2 would take them to H13 after which they planned to walk down. However, Awbie jumped into the water. Later, they iteratively increased the number on their Jump Right piece to figure out how it worked and correctly used three jump blocks in a row in their next session. Halfway through their time in session four, the game forced a hint that asked them to place Walk Up 5. From this point onward, they started combining some of their movements, and they used the numbers more strategically (e.g., Walk Right 2 instead of Walk Right 1 and Walk Right 1), although they sometimes added on an extra command or reused previous lines of code, that resulted in them falling in the water or moving too far.

Duck3 & Duck4. Primarily, Duck3 and Duck4 experienced difficulty counting the spaces, as they frequently double counted both Awbie's initial square and corners during the remaining sessions. For example, they wanted to move Awbie from B5 to F5 and on to F8 but programmed him to Walk Right 5 by counting B5 and then Walk Down 4 by counting F5. Because of the double counting, they often moved Awbie too far. After trying the jump in session 1, they did not use the jump again until session 5 at which point they noticed that the jump skipped a space saying, "...ooo he jumped over." On the second use of the jump in session 5, they used it correctly. On session 6, they used the jump correctly (reinforced by a hint from the game), although they made counting errors (e.g., using Jump Right 4 instead of Jump Right 3).

Duck5 & Duck6. In session 2, Duck5 and Duck6 started using addition facts for larger distances but also made a double counting error. When Awbie was at F16, they said, "We need to go six, but we do not have a number six" to get Awbie to L16. They used an addition fact to solve their problem, stating, "We can have four plus two," but then they noticed they could move another five to Q16. Still looking ahead, Duck6 said they needed to go up five to Q11, but Duck6 double counted Q16, and said it was six. Throughout the other sessions they kept using addition facts to move larger distances and even used the jump block correctly in sessions 2 to 6 to skip a space. By their final session they were seeing distances rather than counting them, immediately making comments such as, "You need a three" when seeing the path.

Discussion and Future Directions

Our analysis presents a fairly positive account of young students learning to coordinate counting, movements, and direction on a grid. Based on our analysis, we noticed that coordination with counting and movement is a multi-dimensional and essential skill which is needed by younger children who partake in programming. Although the students had some struggles with counting (Bofferding et al., 2020; Kocabas et al., 2019, 2021) and moving Awbie in the correct direction, they progressed in counting along the two dimensions and seeing sets of squares in terms of compositions of numbers. These results provide further evidence that programming activities can support mathematical concepts (Fessakis et al., 2013; Friend et al., 2018; Lewis & Shah, 2012).

There were cases where students double counted the starting space and corners and there were times when they under-counted the number of spaces they needed Awbie to move to land on green space or moved Awbie too far. Double counting errors were most common at the beginning of a level, potentially because they had a conflict in terms of how Awbie started or because they got used to Awbie's movements as they moved within a level, resulting in fewer counting conflicts later. Students' under- and over-counts often resulted from misreading the highlighted path showing the potential result of their program, a misinterpretation of a movement block (i.e., thinking the jump or grab blocks would move one space), or their desire to reuse previous code instead of counting again. Although reusing code is efficient, helping students check the feasibility of previous code is also important. The hints provided by the game often led them to change their approach, at least temporarily. In particular, the hints often helped students combine movements so that instead of using several walk blocks, they would use one with a number indicating the number of times they wanted to walk. Parts of the game where the character could move larger distances in one direction also encouraged students to use combinations of numbers, suggesting intentionally designed game features play an important role in both their understanding of the programming commands but also their use of numbers. Some students corrected each other, which helped them progress in their programming. These results suggest that providing students with ways of handling disagreements or talking about their reasoning when programming might be fruitful for encouraging productive peer talk and collaboration.

Acknowledgments

This research was supported by NSF ITEST grant #1759254.

References

Battista, M. T., Clements, D. H., Arnoff, J., Battista, K., & Borrow, C. V. A. (1998). Students' spatial structuring of 2D arrays of squares. Journal for Research in Mathematics Education, 29(5), 503-532.

Battista, M. T. (1999). Fifth graders' enumeration of cubes in 3D arrays: Conceptual progress in an inquiry-based classroom. Journal for Research in Mathematics Education, 30(4), 417-448.

Battista, M. T. (2010). Thoughts on elementary students' reasoning about 3-D arrays of cubes and 306 vanderbilt. In Z. Usiskin, K. Andersen, & N. Zotto (Eds.), Future curricular trends in school algebra and geometry: Proceedings of a conference (pp. 183-199). IAP.

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. Early Childhood Research & Practice, 12(2). Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. Computers & Education, 72, 145-157. https://doi.org/10.1016/j.compedu.2013.10.020

Bofferding, L., Kocabas, S., Aqazade, M., Chen, L., & Haiduc, A. (2020, April 17–21). Exploring practices to support commenting and debugging in early years of tangible programming [structured poster session]. American Educational Research Association Annual Meeting. San Francisco, CA, United States. http://tinyurl.com/yyd7ayh4 (Conference canceled)

Bofferding, L., Kocabas, S., Aqazade, M., Haiduc, A., & Chen, L. (2022). The effect of play and worked examples on first and third graders' creating and debugging of programming algorithms. In A. Ottenbreit-Leftwich & A. Yadav (Eds.), Computational thinking in PreK-5: Empirical evidence for integration and future directions. A Special Research Publication (pp. 19-29). Association for Computing Machinery, Inc. and the Robin Hood Learning + Technology Fund.

Elkin, M., Sullivan, A., & Bers, M. U. (2016). Programming with the KIBO robotics kit in preschool classrooms. Computers in the Schools, 33(3), 169-186. https:// doi.org/10.1080/07380569.2016.1216251

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. Computers & Education, 63, 87-97.

Friend, M., Matthews, M., Winter, V., Love, B., Moisset, D., & Goodwin, I. (2018, February). Bricklayer: Elementary Students Learn Math through Programming and Art. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (pp. 628-633).

Fuson, K. C. (2012). Children's counting and concepts of number. Springer Science & Business Media.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. Early Childhood Education Journal, 41(4), 245-255. https://doi.org/10.1007/s10643-012-0554-5

Kocabas, S., Bofferding, L., Aqazade, M., Haiduc, A., & Chen, L. (2019). Students' directional language and counting on a grid. In Otten, S., Candela, A. G., de Araujo, Z., Haines, C., & Munter, C. (Eds.), Proceedings of the 41st annual conference of the North American Chapter of the International Group for the Psychology of Mathematics Education (pp. 431–432). St. Louis, MO

Kocabas, S., Chen, L., Bofferding, L., Aqazade, M., & Haiduc, A. (2021). Identifying and fixing double counting errors in mathematics and programming. In Olanoff, D., Johnson, K., & Spitzer, S. (Eds), Proceedings of the 43nd annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education (pp. 637–640). Philadelphia, PA

Lewis, C. M., & Shah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (pp. 57-62).

Risley, R., Hodkowski, N. M., & Tzur, R. (2016). Devin's Construction of a Multiplicative Double Counting Scheme: Dual Anticipation of Start and Stop. North American Chapter of the International Group for the Psychology of Mathematics Education.

Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. International Journal of Technology and Design Education, 26(1), 3-20. https://doi.org/10.1007/s10798-015-9304-5