

Mastering Control Structures in Secondary Education: Student Observations and Descriptions of Program Logic

Corinna Mößlacher²[0009-0001-6223-406X], Katharina Brugger³, Tatjana Angermann¹, and Andreas Bollin¹

¹ Department of Informatics Didactics at University of Klagenfurt

² University College of Teacher Education Carinthia

³ Carinthia Education Directorate

`corinna.moessleracher@ph-kaernten.ac.at`

`katharina.brugger@aau.at`

`tatjana.angermann@aau.at`

`andreas.bollin@aau.at`

Abstract. The acquisition of control structures in programming poses a significant challenge for K12 students, often requiring more time than typically allocated in standard lecture schedules. This study uses three distinct experiment groups to investigate the efficacy of different instructional approaches to learning control structures. One group (our baseline) consisted of K12 students with prior programming experience. Another group included novices who received a conventional introduction to control structures. Finally, a third group, also comprised of novices, engaged in an intensive unit employing the "human robot" method, which heavily emphasized control structures. Our findings indicate that even for students with prior experience, mastery of control structures demands extended practice and instructional time. Notably, the "human robot" method significantly enhanced the understanding of control structures among novices, suggesting that more dedicated time and innovative teaching strategies are crucial for effectively teaching these fundamental concepts. Consequently, we recommend that computer science lessons allocate additional time and employ active learning techniques to ensure students develop a robust grasp of control structures.

Keywords: K12 · problem solving · control structures · programming education.

1 Introduction

Mastering control structures is fundamental to learning to program, forming the backbone of logical thinking and algorithm development. Control structures, such as loops and conditionals, are essential for creating efficient and effective programs [4]. However, educators often face the challenge of conveying these concepts within the limited time in standard lecture schedules. Traditional teaching methods may not provide the depth of understanding required

for K12 students to apply these concepts effectively in more complex programming scenarios. Additionally, students new to programming frequently adopt a trial-and-error/tinkering approach [15,6], particularly when using block-based programming languages, which can impede their conceptual understanding and long-term retention of control structures.

In response to these challenges, educational strategies have evolved to include block-based programming [12] and unplugged activities [2,1]. Block-based programming enables students to focus on the semantics and logic of programming without the burden of syntax. Unplugged activities, on the other hand, leverage everyday algorithms, such as planning daily routines or understanding game instructions, to illustrate control structures in a tangible, relatable manner. Despite these innovative approaches, the problem persists: many students struggle to develop a robust understanding of control structures, as evidenced by their performance in subsequent programming tasks.

Given students' persistent difficulties in mastering control structures, our research seeks to identify the most effective instructional intervention for teaching these concepts. Specifically, for the work described in this paper, we aim to determine whether the "human-robot" method (where students take over the role of a robot and get programmed to solve problems by their classmates) and observations during hands-on activities with programmed robots better fosters the learning of control structures.

For this, our study examines the effectiveness of these two instructional strategies through an experimental design involving three distinct groups of students. Group one (G01) includes lower secondary class students with little prior programming experience using Scratch⁴. After a short introduction to the so-called "human robot" (that only included tasks with sequential instruction and no control structures to introduce the concept of an algorithm), they were given the task of observing robots called Bit:Bots⁵. Group two (G02) consists of novices introduced to programming through the "human robot" method, including tasks emphasizing control statements and loops. In contrast, group three (G03), with the most programming experience using BBC micro:bits⁶, had no introduction before analyzing the Bit:Bots.

The paper is structured as follows: We present related work, describe our study with focus on the setting and the different cohorts as well as the observed tasks in the workshop. Then, we present the results of the collected data from the students' worksheets - specifically the differences in the cohorts. In the conclusion, we describe possible impacts on our workshops and introductory programming lessons.

⁴ <https://scratch.mit.edu/> (accessed 7 June 2024)

⁵ <https://4tronix.co.uk/blog/?p=1490> (accessed 7 June 2024)

⁶ <https://microbit.org/> (accessed 7 June 2024)

2 Related Work

Block-based programming environments, such as Scratch and Blockly, have been widely adopted due to their intuitive interfaces, enabling beginners to create programs by snapping together blocks representing different commands. This method has lowered the entry barrier for learning to program and fostered creativity and experimentation [13,12]. Grover and Pea [4] discussed the effectiveness of these environments in developing computational thinking skills among K-12 students, emphasizing the importance of focusing on logic and problem-solving rather than syntax. Studies by Maloney et al. [7] further highlight the benefits of Scratch in making programming accessible to a broader audience, noting its role in engaging students in meaningful computational tasks.

Despite the advantages of block-based environments, students often develop misconceptions about programming concepts and program flow. Pea [11], Sorva [14], Mühling [9], and Du Boulay [3] identified several common misconceptions that learners hold, such as misunderstanding the sequential nature of program execution and the behavior of control structures. Mühling also investigated misconceptions in novice programmers, highlighting the need for targeted interventions to address these misunderstandings. These misconceptions can persist and impede progress in more advanced programming tasks.

Our research aims to fill the gap in understanding the most effective instructional interventions for teaching control structures in programming. While the above-mentioned studies have explored the benefits of block-based environments and identified common misconceptions, there is limited empirical evidence comparing different hands-on and unplugged approaches specifically focused on control structures. By examining the efficacy of the "human robot" method against observations during hands-on activities, our study provides insights into how these different strategies impact the understanding and application of control structures among novice programmers. This comparative analysis contributes to the existing body of knowledge by offering practical recommendations for educators to enhance programming education.

3 Study

As described in the introduction, three cohorts of students, with a total n of 73 students, participated in our study. Four workshops were conducted. The setting differed, but all groups had to do the Bit:Bot activity and complete a worksheet. This section describes the setting and tasks in the workshop, the common core part of the study, the background and information of the three study groups, and the data collection and evaluation process.

3.1 Setting and Tasks

Each group participated in a workshop that introduced Bit:Bot programming.

Before starting programming themselves, all classes participating in the workshop were divided into six groups. Each group received a Bit:Bot robot with an existing program. Their task was to try out the program, describing their behavior in as much detail as possible. No further input was given except for a small description of the robots' sensors and buttons.

Each of the six robots was programmed differently and marked with colored stickers. They also differed in the difficulty of the program. After describing one robot, the students were able to switch robots. The participants were given about thirty minutes for this task. Each group managed to work with two to three Bit:Bots in the allotted time. Table 1 briefly describes each program and its control structures. The tasks came from our repository of Bit:Bot programs that we have used in our workshops for the past 3-4 years.

Color	Task	Control Flow	Difficulty
Blue	The robot is controlled by pressing a button, pauses between commands, and can move and turn	conditional, sequential	Easy
Multi	The robot waits for a button press, turns until a sensor is activated, stops/moves, displays random numbers, stops at number '6'	conditional, sequential	Difficult
Yellow	The robot randomly changes color and direction, with pauses in between	sequential	Medium
Green	Has adjustable counter: robot moves forward and turns 90 degrees when A+B are pressed	conditional, sequential, loops/repetition	Medium
Pink	The robot moves forward, turning left at obstacles	conditional, sequential	Medium
Turquoise	The robot uses a compass, can be rotated manually, and aligns to the north	conditional, sequential	Difficult

Table 1. Overview of the available stations (by color code), their Bit:Bot program, possible observable control flows, and their complexity.

3.2 Cohorts and Workshop

This study focuses on the introduction phase of these workshops.

All cohorts were lower secondary class students from three different schools. This leads to different prior knowledge regarding programming experiences. However, they did not know the "human robot" or Bit:Bots and can be classified as follows (for a summary, table 2 provides an overview of all cohorts):

Cohort G01 started with a twenty-minute introduction to algorithms and sequences of instructions using the "human robot". The groups were divided

and challenged to lay instructions sequences to a specific place on the map. No loops or conditionals were used for G01. After this introduction, G01 received the prepared Bit:Bots for the analysis.

Cohort One (G01)	
School Context:	Two public, lower secondary school classes
Quantity:	35 students in total
Age:	Between 12 and 14 years
Prior Knowledge:	Little to no knowledge about block-based coding with Scratch

Cohort Two (G02)	
School Context:	One private, lower secondary school class
Quantity:	12 students in total
Age:	Between 13 and 14 years
Prior Knowledge:	No knowledge about programming

Cohort Three (G03)	
School Context:	One public, lower secondary school class
Quantity:	21 students in total
Age:	Between 13 and 14 years
Prior Knowledge:	Prior knowledge about block-based programming with micro:bits - event control, conditionals and loops

Table 2. Quick overview of the background of the three different cohorts.

Cohort G02 received a more detailed introduction, lasting about thirty minutes, to algorithms and sequences of instructions using the "human robot". The groups were challenged to lay instruction sequences to a specific place on the map AND to use loops and conditionals. A usage for loops was required to let the robot walk a square. The conditional method was used to examine boxes on the map and to react differently depending on the content of the boxes. After this introduction, G02 received the prepared Bit:Bots for the analysis.

Cohort G03 directly received the prepared Bit:Bots for the analysis without an introduction to the "human robot" since they had the most prior programming knowledge compared to G01 and G02.

Every cohort analyzed existing programs by observing the behavior of the Bit:Bots. However, each group received different input regarding algorithms and unplugged programming beforehand. This leads to the next workshop module because we used the "human robot" for this.

The "human robot" is an unplugged activity for teaching programming and control structures. The material was developed by the computer science lab (Informatik-Werkstatt) at the Department of Informatics Didactics at University of Klagenfurt.⁷

⁷ <https://www.rfdz-informatik.at/informatik-werkstatt/>

In the workshop part of the "human robot," the participants are divided into three different roles. First, the 'programmer' team. This team places predefined instruction cards on a table in consultation. These cards show the commands "one step forward/backward" and "one turn to the left/right." These sequences of instructions later guide a robot through a maze to a predefined destination. The second team, the 'interpreter' team, reads these instructions. Third, the 'robot' executes them. The representation of the "human robot" can be seen as a notional machine [5].

Munasinghe et al. [10] have shown that using unplugged activities similar to the "human robot" as an introduction can facilitate the development of programming skills in students. However, their presented example similar to the "human robot" - the Kidbots by CSUnplugged⁸ - does not include control structures such as conditionals and loops. The "human robot", as in our case, does so. Figure 1 shows an example of a conditional. It should be noted that the conditional, as well as the loop, does not include terms ("if...else" or "for...") used in block-based or text-based coding in our workshops. We have chosen a visual representation so as not to provide any wording that will later be included in the worksheet.

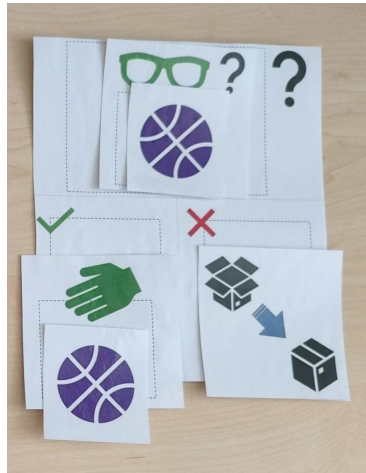


Fig. 1. An example of a conditional that is used in the "human robot"
 A possible written form of this instruction would be: "Do you see the ball?" (Note: in the box you just opened). "If yes: take the ball, if no: close the box".

⁸ <https://www.csunplugged.org/en/topics/kidbots/>

3.3 Data Collection and Classification

To collect the needed data, we printed worksheets with instructions for analyzing the Bit:Bot robots. Although the groups were divided into teams analyzing one robot, the worksheets had to be completed individually and independently of each other. Figure 2 shows the analysis phase of one team in practice.

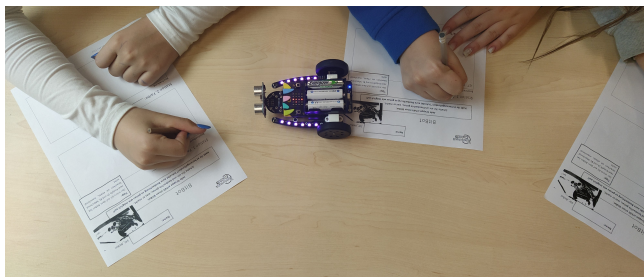


Fig. 2. Three students executing the analysis phase with the multi-colored Bit:Bot

The task was to describe in one's own words what the robot was performing. The students only received information that the robot could drive, with buttons and a distance sensor at the front. If one team finished the analysis, they needed to swap their robots with another. Each team analyzed at least two robots during the phase, which took around thirty minutes. After the workshop, the handwritten description was transcribed in a tabular form by someone not involved in conducting the workshop units.

Every group was recorded in a separate table. Each row is seen as an item and depicts a description of one robot, referenced by its color. Further information about every item is the corresponding child's and their team's index. No personal information was collected at all during the study.

After collecting the data, a classification was carried out. G01 had 99 usable descriptions, G02 had 25 usable descriptions, and G03 had 55 usable descriptions. Of these 179, 24 were selected randomly; a classification was created and applied to 20 other descriptions to test for their suitability. The classification was further improved on this basis. In parallel, ChatGPT4o was used to perform qualitative content analysis, according to Mayring [8]. The proposed classes were analyzed and merged into a common classification. Two researchers completed the classification independently, merged the data in a joint session, and contradictory assessments (about 10 % of the descriptions) were documented or resolved.

Several factors were collected in the study (we were looking at the observation type, the complexity of the description, the technical accuracy, the detection of control structure types, and the area of application), but in this paper, we only look at the factor describing the control structures. They were classified as follows:

Sequential control: Recognition of responses that describe the sequential processing of commands or actions without branches or loops. Distinction between **direct** (*‘first this – then that’*) or **indirect** (enumeration that shows the chronological sequence)

Conditional control: Answers that show an understanding of conditional statements (*if-then logic, selection decisions based on conditions*). Differentiation between **direct** (*‘if-then’*) or **indirect** (when it is implied what happens then; examples: *‘X: right, Y: left’* or *‘at X it drives ...’*)

Loops/repetitions: Responses that describe the recognition of repeatedly performed actions, including infinite loops and conditional loops. Distinction between **direct** (*‘more often’, ‘repeatedly’*) or **indirect** (when it is implied that something happens repeatedly)

Not recognized No control structures have been recognized.

4 Results

We evaluated the results in the context of recognizing conditional statements and loops. Figure 3 shows the detection of conditional statement for G01 whereas Figure 4 respectively Figure 5 shows the results for G02 and G03.

4.1 Conditional Control and Loops

When comparing the cohorts concerning conditional control statements, significant variations in performance were evident.

In group G01, students failed to recognize any control structures in five out of six tasks despite some of these tasks being categorized as difficult.

In contrast, group G02, which underwent the detailed "human robot" preliminary exercise, markedly improved. All students in G02 recognized conditional statements in four tasks, and in two of these tasks, they directly described the target statements.

Group G02 consistently outperformed G01 in every task. The proportion of students failing to recognize statements was lower in G02 across five tasks correctly. The exception was the Turquoise task, where the proportion was slightly higher; however, all recognized statements in this group were direct rather than indirect.

Group G03, which had the most prior programming experience, identified all statements in three tasks. While the proportion of unrecognized statements was relatively low in the remaining tasks, the proportion of direct statements was surprisingly low, sometimes even lower than in G01. This result is unexpected, considering their prior exposure to block-based programming should have facilitated easier formulation of these statements.

Regarding loop recognition, group G02 again excelled, with all students successfully describing loops. In contrast, the other groups had significantly more students who provided no direct or indirect description of a loop.

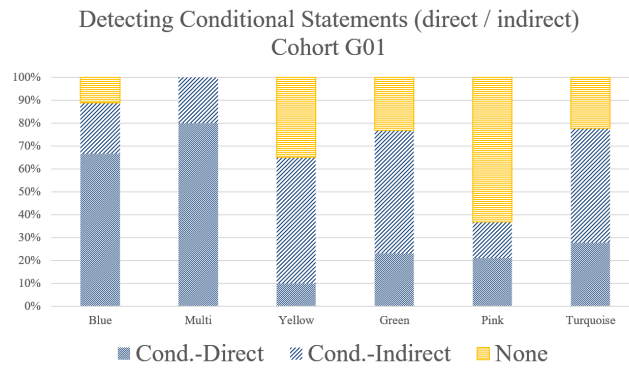


Fig. 3. Relative amount of direct and indirect mentions of control structures of G01 (n = 99 descriptions)

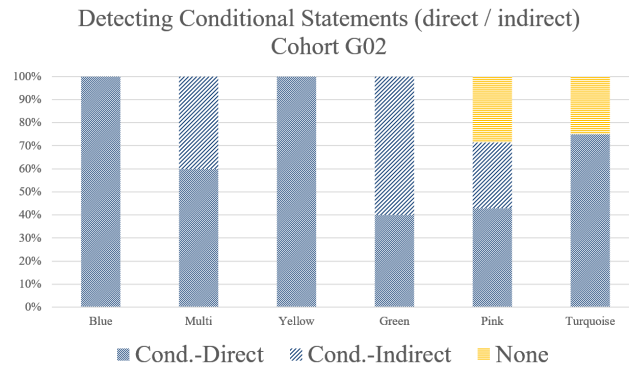


Fig. 4. Relative amount of direct and indirect mentions of control structures of G02 (n = 25 descriptions)

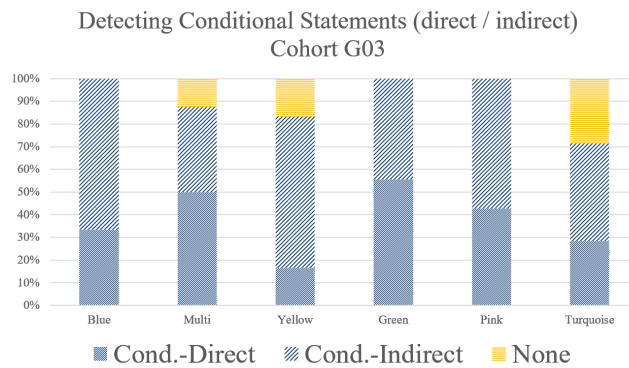


Fig. 5. Relative amount of direct and indirect mentions of control structures of G03 (n = 55 descriptions)

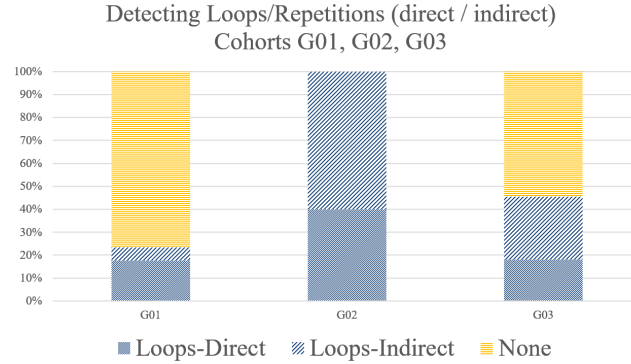


Fig. 6. Relative amount of direct and indirect mentions of loops or repetitions for station "green" for all 3 cohorts (n = 98 descriptions)

4.2 Validity

In the context of our study, several threats to validity must be considered:

Firstly, the cohorts were drawn from different types of schools. Groups G01 and G03 were from public middle schools, whereas G02 was from a private middle school that follows Waldorf education principles. Despite these differences, the educational standards for this age group in the region should ensure comparable writing skills across these cohorts. Therefore, we consider any potential bias due to the school type insignificant.

A more significant concern is the small sample size of cohort G02 compared to the other groups. We could not conduct an additional workshop with another class from the same school to achieve a sample size comparable to G01. Although the small sample size limits the interpretability of G02's results, observable trends allow us to draw preliminary conclusions. In addition, not all students in every cohort analyzed all programs. They were randomly selected to analyze two to three robots each.

Thirdly, we must acknowledge the issue of inter-rater reliability in evaluating qualitative descriptions. Two independent raters assessed each item and compared their evaluations to mitigate this. Discrepancies were minimal, and consensus was reached through discussion. Additionally, a classification run with ChatGPT4 showed no significant deviations, further supporting the reliability of our evaluations.

Finally, the relatively weak results from group G03 may be attributable to the timing of their programming instruction, which did not occur immediately before the survey. This gap may have affected their recall and articulation of programming concepts. Despite expecting that engagement with block-based programming languages would foster a durable understanding, this was not evident in our cohort.

5 Conclusion

Our study aimed to explore the efficacy of different instructional approaches in helping secondary school students recognize and describe control structures in programming. Despite our expectations, the findings indicate significant challenges and variations in students' understanding across cohorts.

The results show that traditional methods and prior experience with block-based programming may not be sufficient to ensure a robust grasp of control structures. Specifically, students in groups G01 and G03, despite their varying backgrounds, exhibited difficulty identifying and articulating control structures accurately. In contrast, group G02, which engaged in the detailed "human robot" preliminary exercise, demonstrated a better understanding, though this was not uniformly strong across all tasks.

These observations underscore the necessity for more innovative and intensive instructional strategies. The "human robot" method appears promising, yet it requires further refinement and consistent application to be fully effective. Additionally, our study highlights the importance of timing and reinforcement in programming education, as gaps between instruction and assessment can negatively impact student performance.

In conclusion, while our research provides valuable insights into the teaching and learning of control structures in programming, it also reveals substantial areas for improvement. Future work should focus on developing and testing more comprehensive educational interventions, ensuring regular reinforcement, and addressing the diverse needs of students. By doing so, we can better equip students with the fundamental programming skills necessary for their academic and professional futures.

References

1. Bell, T.: Cs unplugged or coding classes? *Commun. ACM* **64**(5), 25–27 (apr 2021). <https://doi.org/10.1145/3457195>, <https://doi.org/10.1145/3457195>
2. Bell, T., Witten, I., Fellows, M.: Computer Science Unplugged - an enrichment and extension programme for primary-aged children. *Computer Science Unplugged (csunplugged.org)* (01 2002)
3. Boulay, B.D.: Some difficulties of learning to program. *Journal of Educational Computing Research* **2**(1), 57–73 (1986). <https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9>, <https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9>
4. Grover, S., Pea, R.: Computational thinking in k–12 a review of the state of the field. *Educational Researcher* **42**, 38–43 (02 2013). <https://doi.org/10.3102/0013189X12463051>
5. Kohn, T., Komm, D.: Teaching programming and algorithmic complexity with tangible machines. In: *Informatics in Schools. Fundamentals of Computer Science and Software Engineering: 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018, St. Petersburg, Russia, October 10-12, 2018, Proceedings 11*. pp. 68–83. Springer (2018)

6. Lewis, C.M.: How programming environment shapes perception, learning and goals: logo vs. scratch. In: Proceedings of the 41st ACM Technical Symposium on Computer Science Education. p. 346–350. SIGCSE '10, Association for Computing Machinery, New York, NY, USA (2010). <https://doi.org/10.1145/1734263.1734383>, <https://doi.org/10.1145/1734263.1734383>
7. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* **10**(4) (nov 2010). <https://doi.org/10.1145/1868358.1868363>, <https://doi.org/10.1145/1868358.1868363>
8. Mayring, P.: Qualitative content analysis - theoretical foundation, basic procedures and software solution. Klagenfurt. SSOAR. Open Access Repository (2014)
9. Mühling, A., Ruf, A., Hubwieser, P.: Design and first results of a psychometric test for measuring basic programming abilities. In: Proceedings of the Workshop in Primary and Secondary Computing Education. p. 2–10. WiPSCE '15, Association for Computing Machinery, New York, NY, USA (2015). <https://doi.org/10.1145/2818314.2818320>, <https://doi.org/10.1145/2818314.2818320>
10. Munasinghe, B., Bell, T., Robins, A.: Unplugged activities as a catalyst when teaching introductory programming. *Journal of Pedagogical Research* **7**(2), 56–71 (2023)
11. Pea, R.D.: Language-independent conceptual "bugs" in novice programming. *Journal educational computing research* **2**(1), 25–36 (1986), <https://telearn.hal.science/hal-00190538>
12. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. *Commun. ACM* **52**(11), 60–67 (nov 2009). <https://doi.org/10.1145/1592761.1592779>, <https://doi.org/10.1145/1592761.1592779>
13. Seraj, M., Katterfeldt, E.S., Bub, K., Autexier, S., Drechsler, R.: Scratch and google blockly: How girls' programming skills and attitudes are influenced. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research. Koli Calling '19, Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3364510.3364515>, <https://doi.org/10.1145/3364510.3364515>
14. Sorva, J.: Visual program simulation in introductory programming education. Doctoral dissertation. Ph.D. thesis, Department of Computer Science and Engineering, Aalto University (05 2012)
15. Woo, K., Falloon, G.: Problem solved, but how? an exploratory study into students' problem solving processes in creative coding tasks. *Thinking Skills and Creativity* **46**, 101193 (2022). <https://doi.org/https://doi.org/10.1016/j.tsc.2022.101193>, <https://www.sciencedirect.com/science/article/pii/S1871187122001948>