ISSN 2513-8359

Use Hopscotch to Develop Positive Attitudes Toward Programming for Elementary School Students

Jiahui Wang

Kent State University

DOI: 10.21585/ijcses.v5i1.122

Abstract

With the advancement in technology and the emphasis on computer science education, there has been a strong push for more widespread programming instruction at K-12 and higher education levels. Existing research has mostly focused on students at the secondary and post-secondary levels. Not much research has involved students at the elementary school age, which has been considered a critical age to cultivate an interest in programming. The current study aimed to investigate the effects of a block-based programming interface (e.g., Hopscotch) on elementary school students' attitudes toward programming. In this study, eighteen elementary school students in 4^{th} -5^{th} grades participated in a programming curriculum for about seven weeks in the US. A survey on attitudes toward programming was distributed before and after the curriculum, to explore the change in attitudes toward programming. Students' views about the block-based programming interface (e.g., Hopscotch) were also examined after the curricular activities. Students' activities in lessons and artifacts from the culminating project were observed. The findings indicated that elementary school students had positive views about programming in the block-based programming interface. Also, the block-based programming activities contributed to more positive attitudes toward programming. Implications and limitations of the study were discussed.

Keywords: Computer science, block-based programming, elementary school students, attitudes toward programming, Hopscotch

1. Introduction

1.1 Computer Science Education

United States has an increasing demand for STEM workers than at any time in history, and teaching computer science to students is essential for recruiting STEM workers (Guzdial & Morrison, 2016). However, existing findings indicated that K-12 educators in the United States did not attach enough importance to computer science education (Google Gallup, 2015). Many colleges and universities have also reported declines in enrollment in computer science related courses and majors (Bowman, 2018). As a result, United States has been confronted with a pronounced lack of talents in computer science related professions (Seehorn et al., 2011). The shortage is especially pronounced among females (Grover & Pea, 2013) and non-Asian minorities (Banning & Folkestad, 2012).

1.2 Benefits of Programming Learning

Programming learning has been found to be beneficial to students. According to Wing (2006), the most important skill students can acquire from learning computer science is computational thinking, which refers to the use of abstract thinking to seek a solution to a problem. More specifically, computational thinking is defined as "the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2011, p. 20). Computational thinking has been theorized as a multi-dimensional construct, which encompasses computational concepts, computational practices, and computational perspectives (Brennan & Resnick, 2012). Wing also

ISSN 2513-8359

emphasized the important role computational thinking plays in learning of all subject domains, as she said, "to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (Wing, 2006, p. 33).

In addition to facilitating the development of computational thinking, previous work has also identified some positive effects brought by computational way of thinking, which includes promoting divergent thinking and metacognitive skills (Clements & Gullo, 1984), critical thinking (Clements, & Gullo, 1984; Liao & Bright, 1991), as well as enhancing spatial relations and problem-solving abilities (Fessaki, Gouli, & Mavroudi, 2013; Miller, Kelly, & Kelly, 1988).

1.3 Block-Based Programming Environments

Despite the increasing importance of programming and the benefits of programming learning, computer science education has faced certain challenges, indicated by previous studies (Denner, Werner, & Ortiz, 2012; Robins, Rountree, & Rountree, 2003). Most importantly, programming is a complex mental process and a challenging task (Law, Lee, & Yu, 2010). The conventional text-based environments could burden the learners with syntax and cause frustrations. The learners could lose interest in programming very quickly in these text-based programming environments. To overcome these barriers, various efforts have been made to develop tools and activities to popularize computer science education and engage students in programming activities within K-12 and higher education contexts (Lye & Koh, 2014). With the advancements in learning technologies and their widespread applications in students' learning, several block-based programming interfaces have been designed and introduced to teachers and students to support programming learning at various education levels (Amanullah & Bell, 2020; Denner, Werner, & Ortiz, 2012; Leidl et al., 2017). These block-based interfaces allow the design of programming logics using children-friendly drag-and-drop of blocks. Compared to text-based programming, block-based programming interfaces are useful in reducing the abstractness of syntax and the complexity of programming. They could alleviate frustrations associated with writing abstract syntax and debugging syntax. These programming environments could help novice programmers focus better on the core aspects of programming, that is, the logical thinking process. These environments would allow learners to see how the commands work immediately and detect errors much more easily (Lye & Koh, 2014). Often, the block-based programming interfaces adopt cartoon characters that appeal to the young learners and attract them to construct animations, games, or digital stories relevant to their interests and passions.

Regarding the effects of block-based programming compared to text-based programming, a recent meta-analysis on this topic (Xu et al., 2019) compared block-based programming environments with text-based environments with respect to their impacts on students' cognitive and affective outcomes. With most of the studies examined focusing on secondary education and higher education, the meta-analysis revealed a small effect size favoring block-based programming interfaces on cognitive outcomes; while for affective outcomes, the analysis only identified a trivial effect size. Moreover, the meta-analytic study revealed a significant effect size for elementary school, but only one study focused on the elementary school-age population.

The block-based programming interfaces include but are not limited to Hopscotch, Alice, Scratch, and Minecraft. It has been suggested that students became less burdened by the syntax of programming and became more motivated about programming as exposed to a block-based programming environment named Scratch (Kaucic & Asic, 2011). Block-based programming tools have become an essential component of computer science curricula for high school and university classrooms. Empirical evidence has suggested these block-based programming environments have been proven successful at the secondary level (Burke & Kafai, 2012; Campe et al., 2020; Grover, Pea, & Cooper, 2015; Gunbatar & Karalar, 2018; Meerbaum-Salant, Armoni, & Ben-Ari, 2013; Price & Barnes, 2015) and in higher education (Cetin, 2016; Korkmaz, 2016; Yukselturk & Altiok, 2017). For example, in the study conducted by Burke et al. (2012), researchers explored the effect of Scratch on enhancing middle school students' programming skills. The findings suggested the middle school students were able to master basic programming concepts (e.g., loops, events) to create digital stories. Similarly, another study (Meerbaum-Salant, Armoni, & Ben-Ari, 2013) introduced middle school students to Scratch and the students were found to be able to acquire some important programming concepts, despite having troubles learning some concepts such as repeated execution and variables. Furthermore, Grover and colleagues (2015) introduced middle school students to block-based programming in an introductory programming course. The results revealed that students achieved significant learning gains in algorithmic thinking skills, and they were also able to transfer the knowledge from the block-based programming activities to a text-based programming context. Another study that involved pre-service teachers in a university (Yukselturk & Altiok, 2017) reported that a

ISSN 2513-8359

block-based programming environment (e.g., Scratch) was effective in alleviating negative attitudes such as low self-efficacy among pre-service teachers who came from non-CS backgrounds. Taken together, these studies suggested that exposing learners to block-based programming would be effective in helping young learners acquire basic programing concepts that could be transferrable to text-based programing contexts. It is also reasonable to speculate that the block-based programming tools would be effective in prompting more positive attitudes toward programming among the young learners, due to the fact that the drag-and-drop interface would engage the learners in the logical thinking process while preventing the frustration associated with writing syntax.

1.4 Programming Learning at an Early Age

Developing positive attitudes and stimulating interests in programming at an early age is essential for broadening participation in computer science career (Hainey et al., 2019). Thus, more work focusing on introducing programming at earlier education levels are needed to identify effective approaches and curriculum to facilitate programming education at a young age.

A literature research revealed that the majority of existing research efforts have primarily focused on the contexts of secondary education and higher education, and not so much research has been undertaken on programming education at early age. Among these efforts for young learners, Bers and her colleagues' work has focused on integrating programming in the early childhood classroom (e.g., Bers, 2019, 2020). For example, programming robots have been adopted to support early learning of programming (Kazakoff et al., 2013; Strawhacker & Bers, 2015; Sullivan & Bers, 2013). In one study, Kazakoff and colleagues (2013) found that early learners' sequencing skills could be improved after taking part in a workshop involving the use of programming robots. Moreover, research has been conducted to examine the use of a block-based programming environment - ScratchJr among learners in early childhood (Flannery et al., 2013; Strawhacker & Bers, 2019; Sullivan & Bers, 2019). Specifically, it was suggested that ScratchJr was an effective tool in that the young students (Grade K-2) acquired the foundational programming concepts (Strawhacker & Bers, 2019).

In recent years, growing attention has been given to integrating programming instruction in elementary school (Allsop, 2019; Bell, Duncan, et al., 2016; Bell, Witten, et al., 2016; Duncan et al., 2017). For example, Hainey and colleagues (2019) utilized a novel approach called games-based construction learning (GBCL) to teach programming concepts in upper elementary school. Their findings indicated that the games-based construction learning (GBCL) approach was an effective approach to teach programming concepts in upper elementary school. The elementary school students were able to learn programming concepts effectively. Additionally, other work has focused on methods to evaluate the Computational Thinking (CT) process in an elementary school classroom (Allsop, 2019) and measure students' understanding of computer science concepts (Denner, Werner, & Ortiz, 2012).

With existing evidence showing the effectiveness of block-based programming among students in secondary schools and universities, as well as limited evidence in early childhood education, it is reasonable to expect these block-based interfaces could also benefit elementary school students by involving less complicated programming tasks. It is safe to postulate that block-based programming environments would be useful in stimulating elementary school students' interest, familiarizing them with introductory programming concepts, and eventually building a foundation for text-based programming. In fact, a recent study (Chen et al., 2019) provided some evidence for this assumption. The researchers examined the relationship between undergraduate students' final grades in introductory computer science courses and their very first programming languages before adolescence. The findings revealed that those who received higher final grades in CS courses had their initial exposure to programming in graphical language rather than textual, in or before early adolescent years. The findings suggested graphical language should be adopted for young learners' initial exposure to programming, if programming is to be taught before early adolescence.

A recent meta-analysis, however, indicated that the effects of block-based programming among elementary school students were not adequately studied (Xu et al., 2019). Only a few empirical studies have focused on this population with specific attention given to the cognitive outcomes of block-based programming environments, but findings on the cognitive effects were mixed. For example, adopting a pretest and posttest design, Lai & Yang (2011) examined if block-based programming (e.g., Scratch) would have a positive effect on elementary school students' problem-solving skills and the findings indicated block-based programming activities improved students' problem-solving abilities significantly. On the contrary, Kalelioglu & Gülbahar (2014) failed to replicate the positive effects of block-based programming on problem-solving abilities. Besides examining the

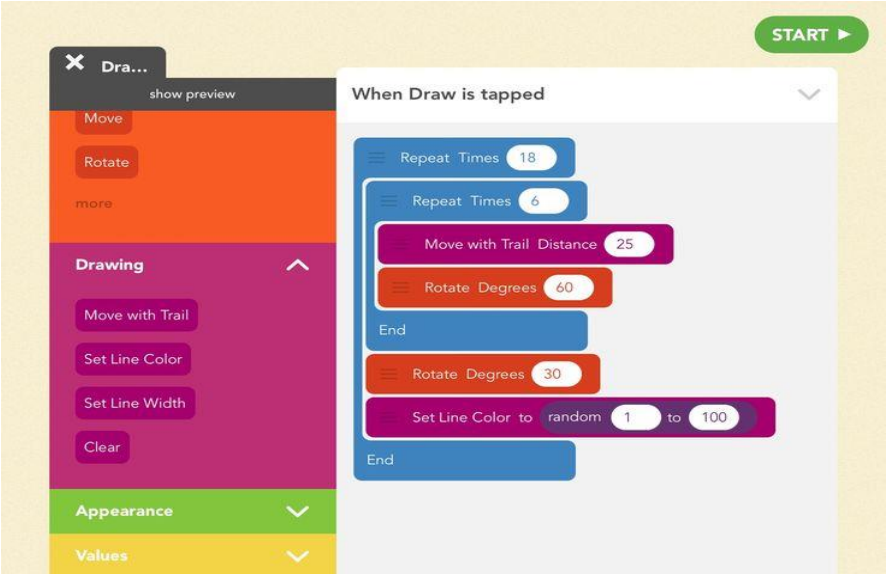
ISSN 2513-8359

influence on problem-solving skills, Sáez-López, Román-González, & Vázquez-Cano (2016) studied the impact of block-based programming on computational thinking and computational practices, and their findings suggested significant improvements in these two areas.

Additionally, the study conducted by Baser (2013) suggested a positive relationship between students' attitudes toward programming and their achievements in programming. It also needs to be pointed out that one of the most significant obstacles in computer science education is the negative attitudes towards programming among students (Bishop-Clark, Courte, & Howard, 2006; Yukselturk & Altiok, 2017). If students could develop early interests in and positive attitudes toward programming, it is more likely they would pursue a major in computer science related disciplines, leading to a career in computer science later on. Thus, it is important to cultivate positive attitudes toward programming, especially at an early age. However, the effects of block-based programming on elementary school students' affective outcomes are not fully understood, either (Xu et al., 2019). Empirically, Duncan & Bell (2015) implemented a programming course for elementary school students aged 11-12. The researchers surveyed the participants to see if the programming course changed their attitudes to computing as career. It was pointed out the study did not measure learners' attitudes prior to the programming course. As all the tests were administered after the course, it was not possible to identify the improvement in attitudes toward programming. More empirical work following a pretest and posttest design is needed to examine the effects of block-based programming on learners' attitudes, especially at the elementary level. Thus, the current study aimed to close the gap in the existing literature by providing data to illustrate the effects of block-based programming on elementary school students' attitudes toward programming.

1.5 Current Study

For teaching programming in elementary classrooms, many tools and resources are available (Duncan, Bell, & Tanimoto, 2014). For example, ScratchJr was developed as an introductory programming environment for young learners aged 5-7, where learners could create a program/story by sequencing different types of programming blocks (e.g., triggering blocks, motion blocks, looks blocks, sound blocks, control blocks, and end blocks, Leidl et al., 2017). In contrast, Hopscotch was designed to target ages 10-15 and was considered as an age-appropriate tool for teaching programming concepts such as loops, randomization, and conditionals to these elementary school students who participated in the current study. Hopscotch was selected also due to its ease of use and compatibility with iPad. Hopscotch, as a typical block-oriented interface, adopts a drag-and-drop graphical programming environment (see a screenshot in Figure 1). Learners don't need to write complicated syntax, and instead, they could construct a programming work (e.g., animation, game, digital stories, etc.) by building a number of blocks in the interface. Hopscotch could introduce novice programmers to fundamental programming concepts such as loops, randomization, and conditional, just to name a few. The current study explored the effects of block-based programming (e.g., Hopscotch) for developing elementary school students' positive attitudes toward programming by adopting a pretest and posttest design.



ISSN 2513-8359

Figure 1. A screenshot of the Hopscotch interface.

The study was designed to address the following two research questions:

RQ1: How does block-based programming experience in Hopscotch influence elementary school students' attitudes toward programming?

RQ2: How do the elementary school students view block-based programming in Hopscotch?

2. Method

2.1 Context & Participants

The study was situated and carried out in a weekend school in the United States. Eighteen elementary school students in grades 4-5 (13 males and 5 females) who had no prior experience in programming participated in the curricular activities. Informed consents were obtained from parents and students. The curriculum comprised seven 1-hour lessons and presented an introduction to elementary programming for this group of students. The curriculum began with an introduction to the field of computer science, which included topics such as careers in computer science, solving problems with computer science, as well as applications of computer science in daily life. The Hopscotch interface was also briefly introduced in the first lesson. Lesson 2 through 6 introduced the students to basic programming concepts such as variables, loops, randomization, and conditionals. Students were first instructed how these concepts could be used in daily life situations to solve real-world problems. The programming instruction was then provided to demonstrate examples and model the process of implementing the commands in the Hopscotch interface. The iPad used by the instructor was projected to the whole classroom to demonstrate the process in the Hopscotch interface. Figure 2 presents an example of how a loop could be accomplished in the Hopscotch interface. Then the students were instructed to exercise the commands and implemented a similar (but not exactly the same) program that incorporated the concepts in the Hopscotch interface on their individual iPads. During the process, the instructor circled around the room and provided scaffolding and feedback for the students as they worked on their own individual programs in Hopscotch. The instructor provided hints and showed how she might approach executing certain commands, instead of giving a direct answer to the problem. The final lesson challenged the elementary school students to self-design a program that incorporated the programming concepts (e.g., variables, loops, randomization, and conditionals) they had learned throughout the first six lessons of the curriculum. For the final project, the students were expected to test and revise the commands to ensure the program could run properly as intended. The final work required the students to apply the previously learned programming concepts in a new context. Each student shared their finished work at the end.

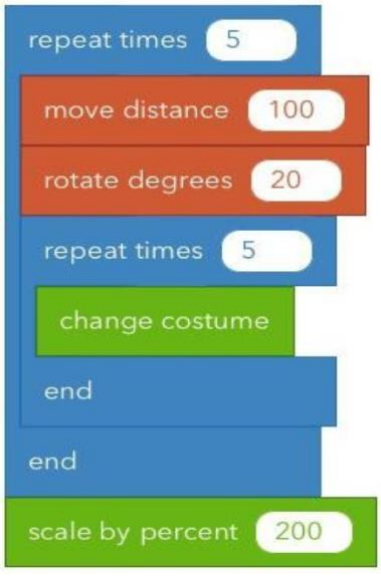


Figure 2. An example of a loop in block-based programming (e.g., Hopscotch).

ISSN 2513-8359

2.2 Data Collection and Measures

In order to understand how block-based programming (e.g., Hopscotch) influences elementary school students' attitudes toward programming, multiple data sources were collected. First, the study adopted a within-subjects pretest and posttest design. A survey was distributed to the students before the class, which included four Likert-scale questions (see Table 1) that were adapted from a previously validated scale on attitudes toward computer science (Hoegh & Moskal, 2009). The survey was readministered to the participants as a posttest after the learning experience in Hopscotch. While the learners responded to the statements, the researcher circled around the room and provided clarifications as needed. In fact, no student raised any questions about the statements. The reliability for the pretest is Cronbach's Alpha α = .645, and the reliability for the posttest is α =.762. On the posttest, students also responded to an additional question, "Programming in Hopscotch is a positive experience for me," on a 5-point Likert scale from *strongly disagree* to *strongly agree*. They were also requested to provide more explanations for their selections. Students' activities throughout the lessons and their artifacts created in the Hopscotch interface were observed.

3. Results

3.1 RQ1: How does block-based programming in Hopscotch influence elementary school students' attitudes toward programming?

To decide the appropriate statistical test to examine if there was a significant difference between the pretest and posttest responses to the four statements that gauged students' attitudes toward programming, data were checked for normality using the Shapiro-Wilk test. The assumption of normal distribution was rejected, which indicated the need to use Wilcoxon signed-rank tests to conduct the analyses. Table 1 presents the descriptive statistics for the level of agreement with each individual statement. The results indicated a significant increase in agreement with the statement after participating in the programming curricular activities. Specifically, for "I will take more programming courses and learn more about programming in future", participants agreed more with the statement after the programming experience, Z = 3.755, p = .000. For "I hope that my future career will involve programming", the results showed a statistically significant difference between pretest and posttest, Z = 3.906, p = .000. For "Having background knowledge and understanding of computer science is valuable in daily life", there was a significant increase in agreement with the statement after the curricular activities, Z = 2.887, p = .004. For "The challenge of solving problems using computer science appeals to me", participants had a significantly higher level of agreement with the statement on the posttest than on the pretest, Z = 3.317, p = .001. For all the statements, participants' levels of agreement with the statements significantly increased from the pretest to the posttest. This result may show that Hopscotch-based programming activities led to more positive attitudes toward programming among elementary school students. Overall, these findings suggested that block-based programming (e.g., Hopscotch) was helpful in getting elementary school students interested in programming and motivated to learn more about computer science in future.

Table 1. Descriptive Statistics for the level of agreement with the statements

	Before (n = 18)		After (n = 18)	
Please circle the number below that indicates how much you agree or disagree with each statement: (1) strongly disagree, (2) disagree, (3) neither agree nor disagree, (4) agree, (5) strongly agree.	*M*	*SD*	*M*	*SD*
S1: I will take more programming courses and learn more about programming in future.	2.94	0.87	4.00	0.49
S2: I hope that my future career will involve programming.	3.11	0.47	4.39	0.61

ISSN 2513-8359

S3: Having background knowledge and understanding of computer science is valuable in daily life.	3.44	0.51	4.00	0.59
S4: The challenge of solving problems using computer science appeals to me.	3.00	0.69	3.61	0.98

3.2 RQ2: How do the elementary school students view block-based programming in Hopscotch?

The results demonstrated that elementary school students' views about block-based programming in Hopscotch were positive after participating in the Hopscotch-based programming curriculum. As students were asked their level of agreement with the statement "Programming in Hopscotch is a positive experience for me", 12 strongly agree with the statement, and 6 agreed with the statement. One participant wrote, "I really enjoyed the app; I would like to learn more about programming." It was also mentioned by two participants, "Hopscotch is easy to use." Another participant expressed her appreciation of the app by noting, "I like the way how the app works; it allows me to see how my codes work right away." Other comments from the participants included the following:

- Programming is fun.
- I like the app.
- Programming is interesting.
- I had lots of fun with programming.
- I like the idea of building animations with my favorite characters.

Overall, the elementary school students felt positive about and engaged in using Hopscotch to practice programming, and the researcher's in-class observations also corroborated this finding. Although the curriculum only involved some basic programming concepts, students learned how to apply abstract thinking to solve a problem (e.g., implement a simple program in Hopscotch). It can be argued that Hopscotch-based programming instruction helped elementary school students comprehend the fundamental programming concepts, which can be gleaned from their final programming products. The final projects students created demonstrated they had mastered the previously learned knowledge and skills throughout the curricular activities (see an example of a student's final project codes in Figure 3). The findings suggested block-based programming activities in Hopscotch were beneficial in assisting the development of programming skills for elementary school students.

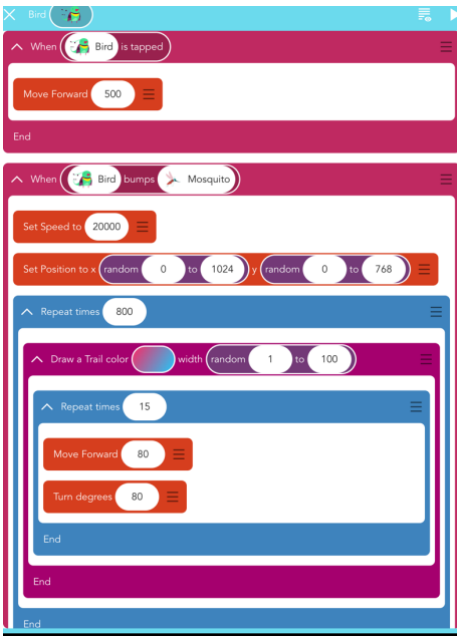


Figure 3. An example of students' final project in Hopscotch.

ISSN 2513-8359

4. Discussion

The study contributed to the effort to expand programming education at the elementary level. The study sought to enhance our understanding of how block-based programming influences elementary school students' attitudes towards programming. The findings revealed block-based programming was effective in cultivating positive attitudes toward programming among elementary school students. It was observed that the students were able to make sense of the programing concepts and were able to apply the concepts to the culminating project. The present results contributed to a growing body of literature seeking to understand the effects of block-based programming at various age levels.

Results further showed participants' views about the block-based programming environment were positive, as evidenced by their responses to the statement "Programming in Hopscotch is a positive experience for me" and the explanations they provided for their selections. Generally, the block-based programming environment was well received by the students, for example, one comment was "I like the app". Participants also thought programming in Hopscotch was a fun experience. For example, three participants respectively commented, "programing is fun", "Programming is interesting", and "I had lots of fun with programming". Another participant expressed similar feeling in the response, "I really enjoyed the app". The fun in the programming experience could possibly be attributed to the fact that the programming interface provided characters that appeal to the young participants, and they can build programs that incorporate these characters. For example, one participant provided support for this speculation and mentioned that "I like the idea of building animations with my favorite characters". Another factor that possibly contributed to the positive views about the Hopscotch programing environment is that the Hopscotch app is easy to use and allows the learners to test the program immediately and fix errors (i.e., debugging and problem solving) as needed. For example, one participant commented, "I like the way how the app works; it allows me to see how my codes work right away". Based on the observation, the students enjoyed the process of building up an animation/program through the app, and a high level of interest and engagement was observed throughout the curricular activities.

This finding is in line with two studies that focused on a different block-based programming environment called Scratch. For example, Sáez-López and colleagues (2016) involved elementary school students in 5^{th}-6^{th} grades in block-based programming activities. Based on students' responses to the questionnaire, students demonstrated positive attitudes toward the block-based programming interface. More recently, Mladenović and colleagues (2017) compared block-based programming (e.g., Scratch) and text-based programming (e.g., Python) for game-based programming among 5th-grade elementary school students. By surveying learners' attitudes toward the programming interfaces after the activities, learners displayed more positive attitudes toward programming to Scratch compared to Python. This observation provided more support for learners' positive attitudes toward the block-based programming activities.

The study yielded several practical implications. The current study demonstrated the possibility of developing positive attitudes toward programming by exposing elementary school students to a programming curriculum involving the use of the Hopscotch block-based programming environment. Teachers and parents of elementary school students should take that into consideration.

Hopscotch-based programming instruction represented the very first exposure to computer programming among the learners who participated in the current study. The results of the study indicated Hopscotch was helpful in developing an early interest in programming among the young learners. The findings provided important implications for computer science educators in the elementary school setting. Next, the study revealed that elementary school students demonstrated interest in using Hopscotch for programming, but teachers' guidance is also important in identifying meaningful curricular activities that are educationally valuable in the Hopscotch interface.

The current study also benefited a broader population of elementary school students and teachers, including students from traditionally underrepresented groups in computer science. The curriculum also provided implications for the design of an effective curriculum to arouse children's early interest in programming and develop programming skills, which will help broaden participation in computer science careers in the long-term.

The findings of the present study should be interpreted in light of several limitations, and future work should seek to address these limitations.

One limitation is that the study adopted a small sample size, and it needs to be acknowledged that the present findings may be limited in generalizability. Future studies could benefit from adopting larger sample sizes to

ISSN 2513-8359

explore the effects of block-based programming activities on attitudes toward programming among young learners.

Second, the study investigated the effects of Hopscotch-based programming instruction on attitudes toward programming. A future study should continue to examine the effects of block-based programming instruction on learners' attitudes toward programming by adopting a more comprehensive scale to measure learners' attitudes.

Another limitation of the study was that the attitude survey was conducted right after the curricular activities. Future research is recommended to adopt a longitudinal design and examine the long-term effects of programming curriculum on learners' attitudes toward programming in the long run.

Previous studies have shown that boys tend to have more positive attitudes toward programming as compared to girls (Baser, 2013; Rubio et al., 2015). These gender differences in students' attitudes towards programming could possibly influence their interests in pursuing computer science majors and eventually undertake careers involving computer science. Future studies could focus on exploring approaches to bridge the gender-based differences in learners' attitudes toward programming.

It is also worth conducting more work to understand the processes of programming learning. For example, a future study could use screen recording and think-aloud protocols (Ericsson & Simon, 1998) to study the processes of programming among elementary school students. The study could examine how novice programmers build and comprehend the block-based codes, which can be used to then improve the learning processes as well as the instructional approaches that target novice learners.

Last but not least, future work is also recommended to examine ways to integrate programming instruction into elementary school STEM learning by infusing computational thinking into the project-based learning of STEM contents.

Funding

The author(s) received no financial support for the research or publication of this article.

References

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30–55.

Amanullah, K., & Bell, T. (2020). Revisiting code smells in block based languages. *ACM International Conference Proceeding Series*, *274*.

Banning, J., & Folkestad, J. E. (2012). STEM education related dissertation abstracts: A bounded qualitative meta-study. *Journal of Science Education and Technology*, *21*(6), 730–741.

Baser, M. (2013). Attitude, gender and achievement in computer programming. *MiddleEast Journal of Scientific Research, 14*(2), 248–255.

Bell, T., Duncan, C., & Atlas, J. (2016). Teacher feedback on delivering computational thinking in primary school. *ACM International Conference Proceeding Series*, *13-15-Octo*, 100–101.

Bell, T., Witten, I. H., & Fellows, M. (2016). *CS Unplugged: An enrichment and extension programme for primary-aged students*. University of Canterbury. CS Education Research Group, New Zealand.

Bers, M. U. (2019). Coding as another language: a pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, *6*(4), 499–528.

Bers, M. U. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.

Bishop-Clark, C., Courte, J., & Howard, E. V. (2006). Programming in pairs with Alice to improve confidence, enjoyment, and achievement. *Journal of Educational Computing Research*, *34*(2), 213–228.

Bowman, D. D. (2018). Declining talent in computer related careers. *Journal of Academic Administration in Higher Education*, *14*(1), 1–4.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vol. 1, 25.

ISSN 2513-8359

Burke, Q., & Kafai, Y. B. (2012). The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. *43rd ACM Technical Symposium on Computer Science Education*, 433–438.

Campe, S., Denner, J., Green, E., & Torres, D. (2020). Pair programming in middle school: variations in interactions and behaviors. *Computer Science Education*, *30*(1), 22–46.

Cetin, I. (2016). Preservice teachers' introduction to computing: exploring utilization of scratch. *Journal of Educational Computing Research*, *54*(7), 997–1021.

Chen, C., Haduong, P., Brennan, K., Sonnert, G., & Sadler, P. (2019). The effects of first programming language on college students' computing attitude and achievement: a comparison of graphical and textual languages. *Computer Science Education*, *29*(1), 23–48.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, *76*(6), 1051.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240–249.

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding? *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 60–69.

Duncan, C., & Bell, T. (2015). A pilot computer science and programming course for primary school students. *ACM International Conference Proceeding Series*, *09-11-Nove*, 39–48.

Duncan, C., Bell, T., & Atlas, J. (2017). What do the teachers think? Introducing computational thinking in the primary school curriculum. *ACM International Conference Proceeding Series*, 65–74.

Ericsson, K. A., & Simon, H. A. (1998). How to study thinking in everyday life: Contrasting think-aloud protocols with descriptions and explanations of thinking. *Mind, Culture, and Activity*, *5*(3), 178–186.

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.

Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. *ACM International Conference Proceeding Series*, 1–10.

Google Gallup. (2015). *Images of computer science: Perceptions among students, parents and educators in the US.*

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199–237.

Gunbatar, M. S., & Karalar, H. (2018). Gender differences in middle school students' attitudes and self-efficacy perceptions towards MBlock programming. *European Journal of Educational Research*, *7*(4), 925–933.

Guzdial, M., & Morrison, B. (2016). Growing computer science education into a STEM education discipline. *Communications of the ACM*, *59*(11), 31–33.

Hainey, T., Baxter, G., & Ford, A. (2019). An evaluation of the introduction of games-based construction learning in upper primary education using a developed game codification scheme for scratch. *Journal of Applied Research in Higher Education*, *12*(3), 377–402.

Hoegh, A., & Moskal, B. M. (2009). Examining science and engineering students' attitudes toward computer science. *39th IEEE Frontiers in Education Conference*, 1–6.

Kalelioglu, F., & Gülbahar, Y. (2014). The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education*, *13*(1), 33–50.

Kaucic, B., & Asic, T. (2011). Improving introductory programming with Scratch? *2011 Proceedings of the 34th International Convention MIPRO*, 1095–1100.

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.

Korkmaz, Ö. (2016). The effect of scratch-based game activities on students' attitudes, self-efficacy and academic achievement. *International Journal of Modern Education and Computer Science*, *8*(1), 16–23.

Lai, A. F., & Yang, S. M. (2011). The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. *2011 International Conference on Electrical and Control Engineering, ICECE 2011 - Proceedings*, 6940–6944.

Law, K. M., Lee, V. C., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, *55*(1), 218–228.

Leidl, K. D., Umaschi-Bers, M., & Mihm, C. (2017). Programming with ScratchJr: A review of the first year of user analytics. *Proceedings of International Conference on Computational Thinking Education*, 116–121.

Liao, Y. K. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research*, *7*(3), 251–268.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, *23*(3), 239–264.

Miller, R. B., Kelly, G. N., & Kelly, J. T. (1988). Effects of Logo computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology*, *13*(4), 348–357.

Mladenović, M., Krpan, D., & Mladenovi, S. (2017). Learning programming from scratch. *Turkish Online Journal of Educational Technology*, *November Special Issue*, 419–427.

Price, T. W., & Barnes, T. (2015). Comparing textual and block interfaces in a novice programming environment. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 91–99.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172.

Rubio, M. A., Romero-Zaliz, R., Mañoso, C., & Angel, P. (2015). Closing the gender gap in an introductory programming course. *Computers & Education*, *82*, 409–420.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129–141.

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., ... & Verno, A. (2011). *CSTA K-12 Computer Science Standards: Revised 2011*.

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, *25*(3), 293–319.

Strawhacker, A., & Bers, M. U. (2019). What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children. In *Educational Technology Research and Development* (Vol. 67, Issue 3). Springer US.

Sullivan, A., & Bers, M. (2019). Computer science education in early childhood: the case of ScratchJr. *Journal of Information Technology Education: Innovations in Practice*, *18*(1), 113–138.

Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners' robotics and programming achievement. *International Journal of Technology and Design Education*, *23*(3), 691–702.

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Wing, J. (2011). Research notebook: Computational thinking-What and why. *The Link Magazine*, *6*.

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study. *Computer Science Education*, *29*(2–3), 177–204.

Yukselturk, E., & Altiok, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology*, *48*(3), 789–801.

ISSN 2513-8359