acrt AI, Computer Science and Robotics Technology

IntechOpen Journals

RESEARCH PAPER

Interaction Patterns During Block-based Programming Activities Predict Computational Thinking: Analysis of the Differences in Gender, Cognitive Load, Spatial Ability, and Programming Proficiency

Abdullahi Yusuf[1,2,*], Norah Md Noor[2] and Marcos Román-González[3]

1 Department of Science Education, Sokoto State University, Nigeria
2 School of Education, Universiti Teknologi Malaysia, Malaysia
3 Faculty of Education, Universidad Nacional de Educación a Distancia, Spain
*Corresponding author. E-mail: abdullahi.yusuf@ssu.edu.ng

Citation
Abdullahi Yusuf, Norah Md Noor and Marcos Román-González (2024), Interaction Patterns During Block-based Programming Activities Predict Computational Thinking: Analysis of the Differences in Gender, Cognitive Load, Spatial Ability, and Programming Proficiency. *AI, Computer Science and Robotics Technology* 3(1), 1–39.

DOI
https://doi.org/10.5772/acrt.36

Received: 17 February 2024
Accepted: 18 June 2024
Published: 26 July 2024

Abstract

The recent advancement in computational thinking (CT) research has reported numerous learning benefits to school-age children. The long-standing perceived difficulty of computer programming has challenged the acquisition of CT skills from programming education. Several block-based programming environments (BBPEs) have been developed to reduce this difficulty and enhance active engagement in computational-related activities. Although numerous studies have examined students' level of interactions during block-based programming modality (BPM) activities, a major gap in the literature is the paucity of research evidence reporting the association between these interactions and CT. This study, therefore, investigates the association between interaction patterns during BPM activities and CT skills. The present study employed a longitudinal approach where the same participants were observed over eight weeks. Thirty-five, second-year-level computer science

and computer education students (mean age: 19.8; male = 23, female = 12) from a research university in Nigeria were recruited. Their computational activities over the study periods were video-recorded. The participants' CT skills were collected using the computational thinking test and the computational thinking scale. Findings indicate four interaction patterns: learner–learner, learner–content, learner–teacher, and learner–distractor. Learner–learner and learner–content were prevalent. The interaction patterns significantly predict CT skills although significant differences exist across gender, cognitive load, spatial ability, and programming proficiency. The research has provided opportunities for educators to integrate BBPEs in learning programming and CT concepts. Although such integration is likely to occur with the help of strong educational policies, teachers are encouraged to cultivate the spirit of collaboration in students during programming activities.

Keywords: computational thinking, block-based programming modality, block-based programming environments, interaction patterns, students

1. Introduction

In recent years, the field of computer programming has assumed a critical role in enhancing diverse cognitive competencies among students [1]. In pursuit of augmenting these cognitive skills, numerous instructional modalities have emerged, broadly classified into block-based (BPM) and text-based (TPM) programming modalities [2]. The BPM presents a visual programming paradigm, employing a metaphor akin to "programming-primitive-as-puzzle-piece," facilitating the conceptualization and execution of computer programming designs [3]. Conversely, TPM adheres to a more traditional approach, necessitating learners' proficiency in coding across diverse text-based programming languages such as Python and Java. Proficiency in these languages holds relevance for undertaking professional programming endeavors and pursuing careers in computer science [4].

Within the programming context, there has been consistent research evidence reporting the close association between programming and computational thinking (CT) [5–9]. In the process, teachers and researchers have put continuous effort toward developing and integrating BPM and TPM into programming education to improve learners' CT skills [2, 10, 11]. On an overarching level, CT encompasses a large intellectual foundation required to provide knowledge and understanding of the computational world, and the ability to employ such knowledge in problem-solving across different disciplines [12–14]. The recent advancement in CT research has reported numerous learning benefits to school-age children, including the ability to solve real-world problems systematically [15, 16]. There is a common agreement that CT permits students to develop creative thinking and problem-solving skills [17].

However, the acquisition of CT skills from programming education has been challenged by the long-standing perceived difficulty of computer programming [15]. To address these challenges, several block-based programming environments (BBPEs), including prominent ones such as Scratch [18] and Alice [19], have been proposed. These environments enable students to interact with programming code through drag-and-drop interfaces, allowing the creation of animations and games without the need for extensive typing. The pedagogical benefits of block-based programs lie in their technological design. For instance, Alice and Scratch minimize syntax errors—a major source of frustration for programming students—by eliminating the need for traditional code typing [20, 21]. This allows students to focus more on understanding programming concepts and logic rather than syntax. Moreover, these environments provide immediate visual feedback, which helps students quickly see the results of their actions and understand the effects of their code. This immediate feedback loop is crucial for learning and helps maintain student engagement and motivation.

However, research evidence has indicated that BPM does not always enhance students' interaction. One explanation is that the visual and interactive elements of these programming environments can overload working memory [22]. This finding is supported by Cognitive Load Theory [23], which suggests that extraneous cognitive load can be imposed by factors such as visual clutter and the complexity of managing numerous blocks. Additionally, understanding and organizing the logical flow of programs in BBPE can be challenging, particularly for beginners. Even though these environments simplify syntax, they still require learners to grasp fundamental programming concepts like loops, conditionals, and variables. Debugging in BPM can also be cognitively demanding as identifying and correcting errors in a visual format requires careful attention and problem-solving skills. Consequently, students often need a high level of spatial ability to effectively navigate and interact with BBPE. The combination of these factors can lead to increased cognitive load, potentially hindering the learning process.

Another plausible explanation is that most BBPEs were developed for novice programmers [24, 25], and the interaction with these programs could lead to an expertise reversal effect, a concept where block-based programs are not beneficial or counterproductive to the learning outcomes of expert programmers [26, 27]. Gender difference is also another reason for the inconsistent benefits of block-based environments. Studies revealed that boys interact more in programming activities than girls [28, 29], and therefore are more likely to interact in BPM activities. However, recent studies found no gender difference in the use of programming environments [30].

Numerous studies have examined students' CT during BPM activities. For example, Bers *et al.* [31] found that students understood the fundamentals of

programming and CT concepts related to sequencing when exposed to BPM. Namli and Aybek [32] examined the effect of BPM on 82 grade-five students' CT. The authors found that BPM activities had a significant effect on CT skills. A recent study that compared students' CT skills in two BBPEs [33] found that augmented reality robotics (AR Bot) enhances algorithm design and algorithm efficiency more than the Scratch environment. Ou Yang's study suggests that although different BPMs could enhance CT skills, novel ones such as robotics are more effective. A more recent study by Sun *et al.* [34] compared Chinese students' programming behaviors, CT skills, and programming attitudes between BPM and TPM. Among their findings, the authors showed that learners in BPM achieved a higher level of CT skills. In the African region, Agbo *et al.* [35] reported that BPM activities significantly increased the number of students who gained CT competency. These findings confirm the widely reported evidence that BPM fosters CT.

However, a major gap in the literature is the paucity of research examining the association between distinct interactions in BPM activities and CT. It is proposed by the researchers that some interactions during BPM activities might foster CT more than other interactions. Nevertheless, to the best of our knowledge, previous literature has not thoroughly investigated how interaction patterns during BPM activities can represent and foster CT as demonstrated in its core principles and practices. In addition to this paucity of research evidence, limited attention is given to how students differ in their CT skills regarding the cognitive load, spatial ability, programming proficiency, and gender when interacting with these environments. A study is needed in this area to enrich the body of literature in the areas of programming and CT. Therefore, the study analyzed students' interaction patterns when exposed to BPM and examined how these patterns predict CT. Specifically, the study addressed the following research questions (RQs):

1. What interaction patterns exist when students are exposed to BPM activities?
2. Is there a significant association between students' interaction patterns and their profiles (gender, cognitive load, spatial ability, and programming proficiency) during BPM activities?
3. What is the level of students' computational thinking skills during BPM activities?
4. Is there a significant association between students' computational thinking and their profiles (gender, cognitive load, spatial ability, and programming proficiency) during BPM activities?
5. Do the interaction patterns predict computational thinking?

2. Literature review

2.1. Definition of computational thinking

The concept of CT has been subject to various definitions and interpretations across the academic literature. Initially proposed by Wing [14] as involving problem-solving, system design, and understanding human behavior through fundamental computer science concepts, subsequent attempts at clarification have led to revised definitions. This definition was refined by Wing [36] to emphasize the thought processes involved in formulating problems and solutions, highlighting the importance of representation for effective execution by information-processing agents. The definition was further clarified by Aho [37], indicating that CT encompasses thought processes for formulating problem-solving steps, resulting in solutions represented as finite sets of computational steps. Other definitions, such as that by Yadav *et al.* [38], emphasize the abstraction of problems and the creation of automatable solutions, while Roman-Gonzalez *et al.* [29] focus on algorithmic problem-solving skills. This study accepts the consensus that CT is rooted in disciplinary concepts of computer science and utilizes computing power. To illustrate this, the following objectives proposed by Fagerlund *et al.* [39] were incorporated:

- **Understanding the Scope and Limitations of Computing:** Participants engaged in discussions and activities exploring the capabilities and constraints of computing systems, such as the finite memory and processing power of computers. Through these exercises, students gained insight into the boundaries and possibilities of computational tools.
- **Understanding the Fundamental Applications of Computers:** Students applied computational tools and models to solve real-world problems in diverse contexts. For instance, they used programming languages like Scratch to create interactive simulations that demonstrated scientific concepts or simulated real-world phenomena.

2.2. The CT framework

Various frameworks have been developed to understand CT. These frameworks include those proposed by Brennan and Resnick [40] (concepts, practices, and perspectives), Repenning *et al.* [41] (abstraction, automation, and analysis), and Shute *et al.* [42] (decomposition, abstraction, algorithm, debugging, iteration, and generalization). Other frameworks include those proposed by Korkmaz *et al.* [43] and Doleck *et al.* [44] (creativity, algorithmic thinking, cooperativity, critical thinking, and problem-solving). Another alternate framework, developed by Çakiroğlu and Çevik [45], recognized abstraction skills as a crucial component of

computational thinking, encompassing subskills such as pattern recognition, generalization, focusing, and elimination within the broader categories of interface, problem-solving, and block structure. The authors proposed that pattern recognition, generalization, focusing, and elimination subskills play active roles in the entire abstraction process.

Although these frameworks have provided indicators for understanding CT, they are criticized for being too limited to specific programming activities and thus insufficient for imparting a comprehensive understanding of programming and CT principles [39]. Recognizing these limitations, a more extensive framework encompassing 14 core educational principles was proposed by Fagerlund *et al.* [39] to provide a broader and more detailed measurement of various CT skills and concepts. Specific indicators include abstraction, algorithm, automation, collaboration, coordination and parallelism, creativity, data, efficiency, iteration, logic, modeling and design, patterns and generalization, problem decomposition, and testing and debugging. Nevertheless, there is still no universally accepted framework for measuring CT.

2.3. Measuring CT

Across the literature, the development of several CT assessment portfolios to evaluate students' CT has been noted. A major drawback of these assessment portfolios is recognized as their closed-access nature (e.g., Fairy Assessment: Werner *et al.* [46]) and their limitation to pre-secondary education (e.g., Chen *et al.* [47]). It has been argued in recent research that the scope of these diagnostic tools is limited and not suitable for providing information on a broad spectrum of proficiency levels [48]. On this account, different open-access tests targeting upper-secondary and university students have been developed.

Among the assessment tests, the two most widely used portfolios are the computational thinking scale (CTS) [43] and the computational thinking test (CTt) [49]. The CTt focuses extensively on "computational concepts" such as sequencing, loops, conditionals, and functions [29]. In contrast, the CTS focuses overly on "computational perspectives," including perspectives on creative thinking, algorithmic thinking, critical thinking, cooperativity, and problem-solving. Because of their comprehensiveness in measuring CT skills, the two tests continue to receive wider applications across different disciplines, involving students of different age groups. Table 1 presents a summary of the design, grounded framework, and target audience of the CT tests.

As previously discussed, CT is measured from a wide range of dimensions. Although it would be difficult to unify these dimensions into a single assessment protocol, several authors strongly recommend the combination of many CT tests to

Table 1. Computational thinking tests.

CT test	Type	Grounded framework/constructs	Target audience	*N*
CTS	Self-assessment test	International Society for Technology in Education framework (ISTE, 2015; creativity, algorithmic thinking, cooperativity, critical thinking, problem-solving)	Undergraduate	29
CTt	Performance test	Brennan and Resnick ([40]; sequences, loops, conditionals, functions, and variables)	Middle school (K-7 and K-8)	28

N, number of items in the test or scale.

measure students' CT skills more extensively [39, 50]. The idea of such unification also includes the reflection of deeper learning that "contributes to a comprehensive picture of students' learning in CT education" [50, p. 2]. To the best of our knowledge, we found a few published empirical studies [50–53] that measure students' CT across different dimensions. This is another gap left in the literature as many studies (e.g., [11, 54–56, 57]) measured CT skills using one dimension, and these studies tend to ignore many CT core principles. In this study, the recommendations of unifying CT dimensions were adhered to. Therefore, the CTS and CTt were used in this study.

2.4. BPM and CT

There is common agreement that programming enhances computational thinking [58]. However, programming difficulties tend to derail the acquisition of CT skills, coupled with the fact that CT concepts are difficult to learn due to their abstract nature [59]. To reduce programming difficulties and facilitate the acquisition of CT skills, several block-based programs were developed, including the prominent ones: Alice and Scratch. Conceived as alternate programming tools during the past 10 years, BPM is generally defined as languages and tools that permit novice programmers to develop software products with little knowledge of the syntax and procedures of conventional programming language [60].

The technological advantage of these programs lies in their drag-and-drop features, which eliminate the hassles of recalling the syntax of conventional programming languages like Java. Since their emergence, they have been applied in many programming education studies to foster CT skills [9, 32, 33]. These studies have demonstrated the positive impact of using block-based programs to enhance students' CT skills, including executive functions that help them to solve and analyze problems.

For example, it was found by Bers *et al.* [31] that the fundamentals of programming and CT concepts related to sequencing were understood by students

when exposed to BPM. The effect of BPM on 82 grade-five students' CT was examined by Namli and Aybek [32]. It was found by the authors that CT skills were significantly affected by BPM activities. In a recent study comparing students' CT skills in two BBPEs [33], it was found that AR Bot enhances algorithm design and algorithm efficiency more than the Scratch environment. Ou Yang's study suggests that although different BPMs could enhance CT skills, more effective results are achieved by novel ones such as robotics.

2.5. Interaction patterns during BPM activities

Across the literature, students' interaction in BPM activities is largely conceived as engagement. The two terms are often used interchangeably. We argue that engagement is a broader term used to represent vision (cognitive), action (behavioral), and emotions (affective). Interaction on the other hand represents specific activities that overly reflect the cognitive and behavioral components of engagement. In a typical classroom activity, authors proposed that such engagement represents three main interactions: interaction with course content, teacher, and peers [61, 62]. The first pattern is termed "learner–content interaction," which occurs when students are watching an animated programming video or performing a programming task. The second pattern is "learner–teacher interaction," which occurs when learners are asking or responding to teacher questions. The third pattern is "learner–learner interaction," which occurs when learners engage in some kind of collaborative task such as think–pair–share. Because students often exhibit different attention-related behaviors in a typical classroom [63], we propose a fourth interaction pattern known as "learner–distractor interaction," which occurs when learners are interacting with unauthorized devices such as mobile phones or engaging in off-point discussion.

Studies have explored different interactions in the context of BPM activities. A recent study by Hopcan *et al.* [64] investigates the interaction sequences of 14 pre-service teachers. The study found significant behavioral sequences that include the development of algorithms, proposing the next steps to be taken, and waiting for approval. Collaborative activities were more predominant in male students. Another study by Olsson and Granberg [65] observed the importance of student–teacher interaction in a Scratch environment to solve mathematical problems. The study found that "well-prepared general and task-specific questions help students to overcome difficulties of learning mathematics while programming" (p. 1). Overall, studies examining interaction patterns during BPM activities are numerous, and they generally yield positive results. However, we found none that predicted computational thinking skills using these patterns. This is a departure point of the present research from prior studies.

2.6. Factors influencing classroom interaction in BPM activities

Despite the learning benefits of BPM, research examining their pedagogical effectiveness has shown that they are not always effective (for a review, see [22]). Thus, interactions with these programming tools are determined by external variables that include cognitive load level, spatial ability, programming proficiency, and gender.

2.6.1. Cognitive load

Few studies have shown that some BBPEs can sometimes overwhelm students' working memory, leading to cognitive overload. Although these tools aim to enhance engagement and reduce programming difficulty, the excessive use of blocks for debugging codes within the environment may distract learners and impede their ability to focus on the environment. Additionally, some studies have highlighted challenges associated with transitioning from block-based programming to text-based coding [51, 66]. While block-based environments provide a visual and intuitive entry point to programming, students may encounter difficulties when transitioning to traditional coding languages due to differences in syntax and structure.

2.6.2. Spatial ability

Spatial ability is generally conceived as a type of cognitive function that is essential in processing visual information [67]. The predictive effect of spatial ability on interaction with BPM is explained by two hypotheses: the ability-as-compensator hypothesis [68, 69] and the enhancer hypothesis [68, 70]. The ability-as-compensator hypothesis assumes that BBPE can benefit students with low spatial abilities by reducing the mental effort required to work with programming blocks and illustrations. On the other hand, students with high spatial abilities do not benefit from these tools because they already have the cognitive functions required to generate sufficient mental representation regardless of the presentation formats [67].

The enhancer hypothesis suggests the opposite and claims that high spatial ability learners benefit more from dynamic visualization compared to low spatial ability learners. Recent research has confirmed the validity of the enhancer hypothesis by indicating that block-based visualization was more beneficial to learners with high spatial abilities [67]. One previous study also found that the integration of 3D program visualization enhances visual interaction only in students with high spatial ability [70].

2.6.3. Programming proficiency

An overwhelming number of research studies have indicated that BBPEs were specifically developed for novice programmers [71, 72]. Therefore, exposing expert

programmers to these tools might lead to an "expertise reversal effect," a concept where block-based environments become counterproductive to the learning outcomes of expert programmers [27]. However, as put forward by a recent study [26], the expertise reversal effect also occurs with a high level of expertise. In this situation, highly interactive BBPEs, including those that were designed to enhance deep exploration of concepts, may overload the memory of novice learners and benefit the expert learners.

2.6.4. Gender

Research on gender differences concerning the interaction with BPM activities has attracted controversies. There is no widely accepted evidence suggesting that interaction with BBPEs is in favor of a particular gender category. For example, several studies have reported that males interacted more (e.g., [28, 29]) and several others have reported females (e.g., [73]). There are also quite a few others that reported no difference [74, 75]. At this point, gender differences in the interaction with BPM activities remain a subject of continuing debate.

2.7. Factors influencing CT skills

Ideally, any factor that influences students' interaction within BBPEs should also influence the acquisition of CT skills. This is based on the assumption that students' skills in BBPEs are significantly correlated with their CT skills [5, 6, 74]. Despite this assumption, prior studies often examined differences in gender and programming experience [11, 72, 74] with contradicting results. Only a few studies investigated the differences or relationships between computational thinking and cognitive load or spatial ability (e.g., [51, 66]).

3. Methodology

A longitudinal approach was employed in the present study, where the same participants were observed over eight weeks to examine their patterns of interaction during BPM activities.Studies examining classroom interaction patterns often employed video data as their primary source of information. However, the main challenge of such observation is the rigorous task involved in coding a large amount of observation due to repeated viewings [76]. An alternate approach, which requires classifying classroom interactions and interpreting their educational relevance, was employed. Prior studies have shown the possibilities of generating sufficient streams of patterns using this observational approach. It is expected that our analysis will produce interaction patterns that support theories and practice and contribute to the ongoing refinement of classroom engagement. Figure 1 illustrates the overall data collection process.

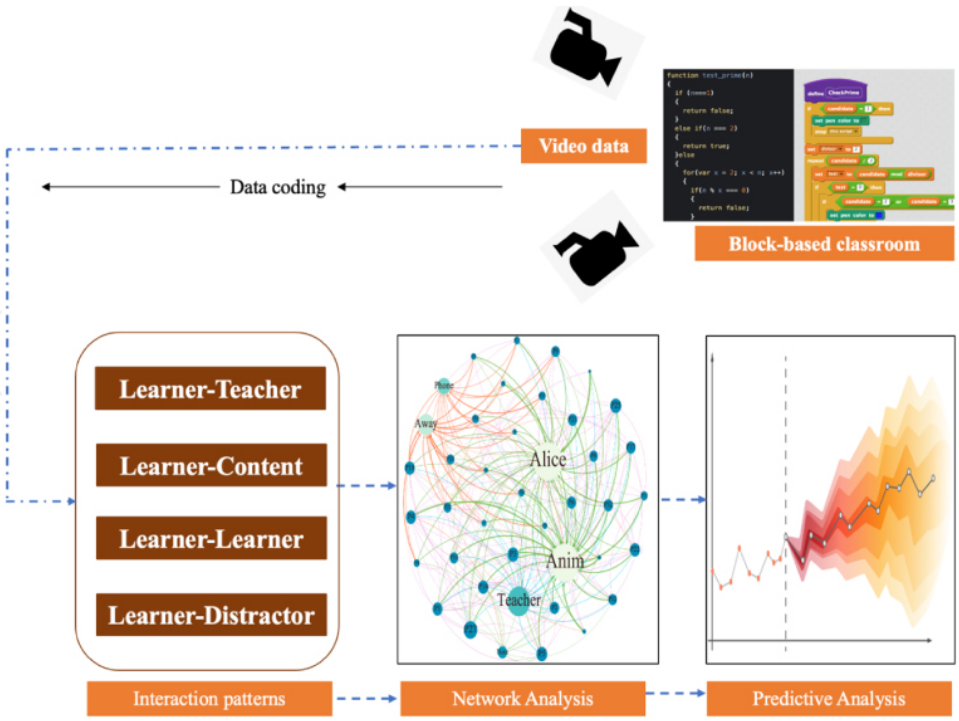

Figure 1. Data collection process.

3.1. Participant recruitment

Thirty-five, second-year-level computer science and computer education students (mean age: 19.8; male = 23, female = 12) from a research university in Nigeria were recruited. Across the literature, there is no numeric standard for sample size adequacy in studies involving video analysis. However, several authors recruited a minimum of 30 participants due to the strenuous task involved in repeated viewing and coding of video data [76, 77]. Recruitment of the participants was carried out using stratified random sampling. Stratification was done to ensure that the two departments (computer science and computer science education) have a representative sample, while randomization was done to ensure that each student from the two strata stands an equal chance of being selected [78].

The research benefit of this sampling technique is that individual characteristics are equally distributed by chance compared to other sampling strategies where such control is limited. Creswell [78] contends that random assignment prevents selection bias that may arise from the personal characteristics of the participants. Although the participants reported not being formally exposed to core programming concepts (evidence from preliminary self-report), initial analysis of their programming proficiency indicated that quite a few are expert and intermediate programmers while a large proportion were novice programmers. It should be noted in this

research that the term "expertise" is conceived as a narrow, task-specific proficiency rather than genuine high-level professional expertise. The differences in expertise might be a result of knowledge gained from a first-year course, "Introduction to Problem-solving," where all computer science and computer education students were exposed to theoretical computing and algorithmic thinking skills.

3.2. Research context

The study occurred in a formal classroom setting where the participants first interacted with a 20-min animated video that teaches computational concepts (see Figure 2) and later participated in a practical session that exposed them to Alice programming across eight weeks. As discussed in the literature section, learning computational thinking is crucial to solving programming and general problems. However, an important but less discussed issue is the concepts of CT to be taught in schools and the instructional environment to be used without significantly changing the existing curricula of computing education. Some authors have proposed important CT concepts to be taught. However, one useful proposal in the context of programming is that proposed by Roman-Gonzalez [49] based on Brennan and Resnick's [40] framework, including sequencing, conditionals, looping, and functions.

Figure 2. Alice 3D animated video demonstrating practical examples of loops.

Recognizing the importance of interactive multimedia instruction, a 3D animated video package that teaches CT concepts was developed using Alice (version 3).

Although Alice is not commonly used in studies measuring CT, its selection was deliberate due to its unique advantages (for a review, see [22]). According to Yusuf *et al.* [77], Alice offers a visually intuitive and user-friendly interface, making it accessible to novice programmers. Additionally, its block-based approach simplifies complex programming concepts, facilitating engagement and understanding among participants [19]. The Alice package was validated based on an expert review of two senior lecturers who specialized in animation development. The CT concepts proposed by Roman-Gonzalez [49] were included in the animated video along with introductory concepts. Overall, six CT concepts were included in the animated video: variables, data types, sequencing, conditionals, looping, and functions. These concepts were taught across the eight weeks with each session being supplemented by a 30-min practical session in the laboratory. Figures S1–S4 in the Appendix illustrate examples of practical tasks performed by the students during the practical sessions.

Across the eight weeks of the intervention, animated instructions took place in an open classroom while practical sessions took place in a computer laboratory setting where each participant interacted with the Alice software from their individual workstation. Information from the animated CT concepts and the practical tasks were delivered via an electronic board across the eight weeks. All programming activities were video-recorded to analyze the classroom interaction patterns. To obtain clear footage, three video cameras were positioned in front of the class—one at the center and two on the sides—facing the students. The cameras were mounted on tripod stands, which were adjusted to a height of 1.43 m above the ground. To ensure high-quality recordings, all cameras were set to a 0.39x wide angle, 3x digital zoom, and 4 K resolution.

The importance of obtaining explicit consent from participants during overt observation is acknowledged in this research. Therefore, before the intervention, an information sheet was sent to the participants, which included details about the video recordings. Additionally, this information was reiterated to the participants every day of the intervention. Participants were informed that their participation was voluntary and that they could withdraw at any time and request the deletion of their video data. Overall, the study was approved by an institutional review board.

3.3. BPM tasks

Four programming tasks, related to the four CT concepts, namely, sequencing, conditionals, looping, and functions, were used for developing students' CT skills in the Alice environment (see sample in S1–S4 in the Appendix). For each CT concept, the participants were presented with visual information that illustrates the behavior of a character in Alice and then instructed to write down the corresponding pseudocodes. Then they were instructed to implement the code in Alice to mimic the

exact behavior of the character. For example, on sequencing, an Alice video containing a "human Biped" was presented to the participants. The Biped's behavior includes rolling its head from left to right in sequence, with a delay of 0.25 s, and returning the head to its original position. Although participants performed their programming tasks on individual workstations, they could seek help from the instructor (researchers), colleagues, and the lesson notes given to them. On completion of each task, the participants shared their screens on the electronic board, where they received immediate feedback and recognition for a job well done.

3.4. Materials

Six data collection instruments were used in this study, including video recording devices that capture overall students' programming activities, a CTt [49] that collects students' CT skills based on concepts, and a CTS [43] that collects students' CT skills based on perspectives. Others include cognitive load test, spatial ability test, and programming proficiency test (PPT).

3.4.1. Computational thinking test

The CTt is a 28-item performance test developed from the framework of Brennan and Resnick [40]. The test overly measures CT concepts and ignores CT practices and perspectives. Computational concepts addressed in the CTt include basic directions and sequences, loops, conditionals, and functions. Post-validation of the test [29] revealed a reliability coefficient of 0.793 for the entire sample: 0.721 for 5th and 6th grades, 0.762 for 7th and 8th grades, and 0.824 for 9th and 10th grades. Although the instrument was validated on middle-school students, it has been successfully applied to college and university students (e.g., [51, 79]), with substantial validity and reliability. Test samples are presented in Figures 3 and 4.

3.4.2. Computational thinking scale

The CTS is a 29-item self-assessment test developed from the framework of the International Society for Technology in Education (ISTE, 2015). The instrument extensively measures CT skills from students' perspectives. Constructs assessed include **creative thinking** (e.g., *I believe that I can solve most of the problems I face if I have a sufficient amount of time and if I show effort*), **algorithmic thinking** (e.g., *I can immediately establish the equity that will give the solution of a problem*), **critical thinking** (e.g., *I am good at preparing regular plans regarding the solution of complex problems*), **cooperativity** (e.g., *I like solving problems related to group projects together with my friends in cooperative learning*), and **problem-solving** (e.g., *I have problems in demonstrating the solution to a problem in my mind*). All items were rated on a 5-point scale (1 = never, 5 = always). The scale achieved a 0.822 reliability coefficient for the

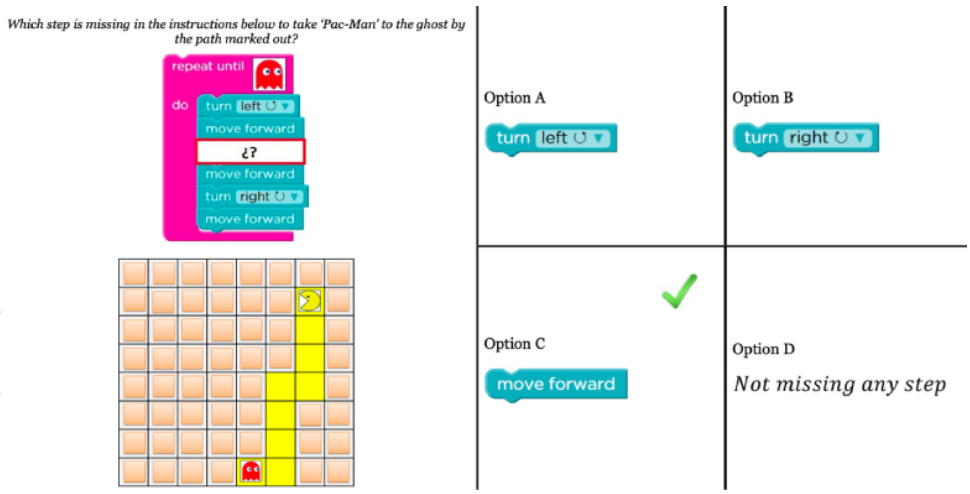

Figure 3. Task: repeat until [49].

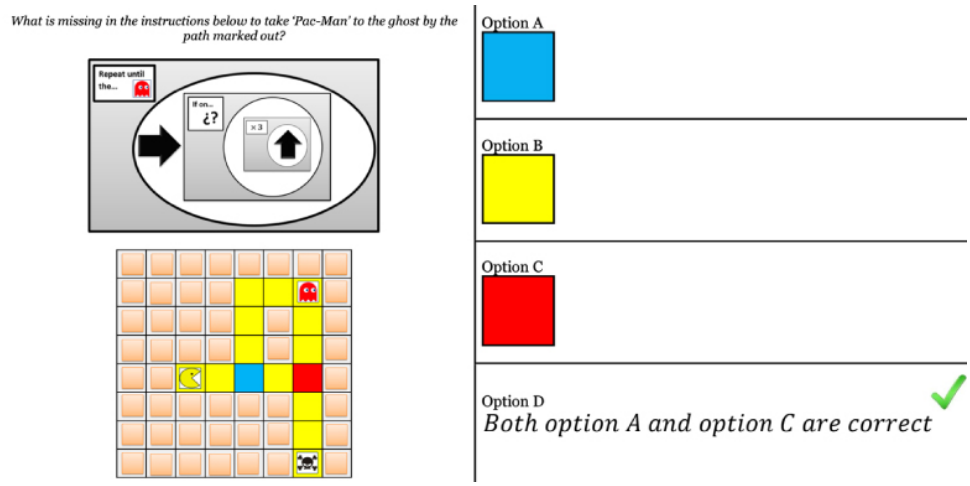

Figure 4. Task: directions and sequences [49].

overall items: 0.843 for creativity, 0.869 for algorithmic thinking, 0.784 for critical thinking, 0.865 for cooperativity, and 0.727 for problem-solving. Table 2 highlights the measured computational concepts and perspectives.

3.4.3. Cognitive load test

Although critics have pointed out difficulties involved in quantifying cognitive overload and have suggested impossibilities for its measurement, researchers have employed several subjective ratings in measuring cognitive load [80–82]. These ratings have provided novel approaches and have been employed in several studies. For this reason, participants' cognitive load was measured by a combination of two subjective ratings. The first item requires the participants to indicate their level of

Table 2. CT concepts and perspectives.

	CT concepts	CT perspectives
Sequence	X	O
Loops	X	O
Conditionals	X	O
Functions	X	O
Creative thinking	O	X
Algorithmic thinking	O	X
Critical thinking	O	X
Cooperativity	O	X
Problem-solving	O	X

X = measured, O = not measured.

perceived difficulty ("How easy or difficult was it for you to work on this task"; 1 = very easy, 9 = very difficult; Kalyuga *et al.* [80]), while the second item requires them to indicate their invested mental effort ("How much mental effort did you invest to work on this task"; 1 = very low, 9 = very high; Paas [81]).

3.4.4. Spatial ability test

The participants' spatial ability was measured using the paper folding test (PFT) developed by Ekstrom *et al.* [83]. The PFT comprises 20 items, with each item consisting of 2 to 4 images illustrating the process of folding a piece of paper. On completion, the folding process illustrates where a hole was punched through a visible circle. Each folded paper is accompanied by five images of unfolded papers with several punched holes. From this visual information, participants were requested to point to an image that correctly displayed the unfolded paper with the correct punched holes. Each correct answer attracts 1 point, yielding a maximum of 20 possible points.

3.4.5. Programming proficiency test

The participants' programming proficiency was measured using a PPT based on Java programming. The test is a departmental test used for formative assessment in the 2019/2020 academic session. This test was used because it had been internally validated by two experts and subsequently used as a proficiency test for students. The test consists of 10 questions. The first five questions require syntax recognition; the next four questions are on output identification; the last question requires participants to write a simple program that displays even numbers from 2 to 100.

3.5. Reliability analysis

A pilot trial was conducted to administer the computational thinking, cognitive load, and spatial ability tests to 100 undergraduate computer science students (not among

the study participants). Their participation in the pilot study was voluntary. Data collected from the pilot study was subjected to reliability analysis to establish the suitability of the adopted instruments for the study. Kuder–Richardson 20 (KR-20) was conducted on the data collected from the CTt and PFT. In contrast, Kuder-Richardson 21 (KR-21) was conducted on the data collected from the CTS and cognitive load test. KR-20 is a measure of reliability for tests with binary responses (i.e., tests with right or wrong responses) having varying levels of difficulty. KR-21 on the other hand is a reliability measure for identifying the internal consistency of tests with partial credit responses, such as the Likert scale. The reliability analysis revealed a reliability coefficient of 0.846 for the CTt, 0.855 for the CTS, 0.762 for the cognitive load test, and 0.824 for the spatial ability test. Details of the reliability analysis are presented in Table 3.

3.6. Data and procedure

The participants reported no prior experience with Alice. For this reason, a 1-h seminar was conducted to expose them to the software before the intervention. The rationale behind this was to make the students familiar with the Alice environment and how characters can be added to and manipulated in the virtual world. The participants' activities were recorded after informed consent had been obtained. For every intervention, the video footage of the animated instruction and the corresponding practical session were combined, yielding eight video files. Each video file lasts approximately 50 min. During the analysis of video recordings, a manual coding approach was performed by two experts. This involved carefully observing the video footage to identify cues indicative of different interaction patterns, including gaze directions and participants' classroom activities. To evaluate the faces of pupils, the experts closely examined the video recordings to

Table 3. Reliability analysis.

	No. of items	Reliability
Concept	28	0.846
Sequence	4	0.823
Loops	8	0.847
Conditionals	12	0.822
Functions	4	0.893
Perspective	29	0.855
Creative thinking	8	0.895
Algorithmic thinking	6	0.838
Critical thinking	5	0.875
Cooperativity	4	0.793
Problem-solving	6	0.874
Cognitive load	2	0.762
Spatial ability	20	0.824

identify instances where participants displayed behaviors such as gaze direction, engagement in classroom activities, and disruptive incidents, which served as indicators of distinct interaction patterns. This process involved annotating specific moments in the video where interactions occurred and paying close attention to the activities and participants' language.

Self-referential evaluation techniques were not employed in this study. This evaluation typically involves participants reflecting on their own behaviors or interactions. Although this approach can provide valuable insights, it was not a focus of the analysis. Instead, the emphasis was on objectively capturing and categorizing observable interaction patterns among participants without direct involvement or input from the participants themselves. Overall, using the manual coding approach, the authors ensured thorough and detailed analysis of the video data, allowing for comprehensive identification and categorization of interaction patterns among participants.

The data coding involves a systematic procedure. First, the video data was coded using a researcher-coded approach [77]. The coding commenced by identifying distinct patterns during BPM activities [84, 85]. Four interaction patterns were coded: learner–teacher, learner–content, learner–learner, and learner–distractor interactions. Within the learner–teacher interaction domain, participants can request the teacher's attention, initiate a talk with the teacher, or ask for more explanation of difficult programming concepts. Within the learner–content interaction, the participants can watch the animation/perform programming tasks in Alice, or read a programmed lecture note given to them. Within the domain of learner–learner interaction, the participants have many options: they can seek help from a colleague, engage in pair or collaborative activities such as meaningful discussion and pair programming, or think and share their ideas with others. Within the domain of learner–distractor interaction, the participants can engage in non-classroom activities such as checking their phones, looking outside the classroom, or engaging in off-point discussion with others (here not categorized as learner–learner interaction). More explanation of the interaction indicators is provided in Table 4. As researchers, control over the participants regarding their preferred interaction was not exercised. Instead, flexible choices were given to the participants to transition or remain within their interaction state.

The classification of interaction patterns is exhaustive because there are no other interaction indicators present in the literature. Each of the observed patterns was categorized in a mutually exclusive way. For example, a participant cannot interact with the content and at the same with his or her peers. Goldberg *et al.* [63] strongly recommend this coding pattern to avoid overlap. To optimize the use of a large dataset, 20 min of video time from the video footage in each intervention was extracted, and the interaction patterns were observed using 5-second intervals.

Table 4. Coding scheme of interaction patterns in a BPM classroom.

Interaction patterns	Numeric code	Indicator
Learner–teacher	1	Student calls for teacher's attention, asks questions, or initiates talks with the teacher.
Learner–content	2	Student watches the animation, takes notes, and works with Alice software to perform programming tasks.
Learner–learner	3	Voluntarily seeks help from a colleague or engages in pair or collaborative activity such as meaningful discussion, peer programming, or think–pair–share.
Learner–distractor	4	Engages in activities not related to the first three categories, e.g., interacting with phones or looking outside the classroom.

Note: Adapted from Kumar *et al.* [61].

Although interval coding may lack accuracy, it synchronizes separate streams of learning patterns [76]. Previously, Andrade *et al.* [76] used a 10-second interval, but the authors believe that 5-second intervals are sufficient to capture information about students' interactions. The data matrix contains 67,200 observations from the four interaction patterns across the eight interventions.

As reported earlier, the coding of the video data was done by two independent coders. The experts were requested to follow the coding template strictly. The coders could also report the limitations of the coding template in the course of coding the interaction patterns. The degree of agreement between the coders was calculated using Cohen's kappa [86] coefficient (κ) in IBM SPSS version 23. Cohen's kappa measures the degree of agreement between raters. Values of 0.4–0.6 indicate fair reliability, 0.6–0.75 indicate good reliability, and values above 0.75 indicate excellent reliability. In this study, the degree of agreement between the two coders is $\kappa = 0.87$, suggesting the suitability of the behavioral features.

Before the interventions, the PPT was administered to the participants and their scores were recorded in an Excel worksheet. The first nine questions have a maximum score of 9 points while the 10th question has a maximum score of 6 points, leading to a total of a maximum of 15 points. After the interventions, the computational thinking, cognitive load, and spatial ability tests were administered via Google Forms. To establish independent responses, the participants were placed in a formal classroom setting while intensive supervision was conducted.

3.7. Data analysis

Various analytical tools were employed to address the RQs. First, a network analysis in Gephi software was employed to investigate the interaction patterns that exist

during the interventions (RQ1). Network analysis measures social structures and relationships across different entities [87]. In a typical network graph, two important entities exist. The first entity includes the points (called nodes), and the second entity includes the lines that connect the points (called edges). In this study, nodes correspond to specific patterns of interaction observed among participants during the interventions. These nodes serve as the building blocks of the network, highlighting the various forms of engagement within the classroom setting. Edges, on the other hand, represent the connections or relationships between nodes in the network. They are depicted as lines that link pairs of nodes, indicating the presence of interactions or associations between them. Edges also signify the occurrence of transitions or shifts between different interaction patterns. For example, an edge between a learner–teacher interaction node and a learner–content interaction node indicates a transition from engaging with the teacher to interacting with instructional materials.

The network graph involves a multipartite network model that depicts the co-occurrence patterns among students during the learning process. The presence of 41 nodes was noted: teacher, animated instruction (Anim), Alice software (Alice), note, phone, away, and the 35 participants. Each participant served as a unique node that interacted with the teacher, animation, Alice software, notes, phone, outside classroom environment, and other students.

To address RQ2, the interaction patterns across different factors were estimated using the network model. The significance of these interactions was examined using the chi-square test of independence. Descriptive statistics, including mean and standard deviation (SD), was employed to measure the participants' scores from the computational thinking tests (RQ3). Differences in the CT scores were examined using independent samples *t*-test for gender and analysis of variance for other factors (RQ4). Before this, preliminary assumption tests were conducted to check for homogeneity of variance and the presence of outliers, with no violations noticed. Lastly, the participants' interaction data was used to predict their CT skills (RQ5). To address this, the ordinal logistic regression (OLR) and binary logistic regression (BLR) were employed. The use of OLR and BLR depends on the measurement scale of the response variable. The OLR is used when the response variable is measured in an ordinal scale (e.g., the CTS). In contrast, the BLR is used when the response variable is measured in a binary scale (e.g., the CTt). However, both tests give an odds ratio (OR) of prediction.

4. Results

Preliminary results (Table 5) show that, on average, the participants had a low cognitive load (M = 2.89 ± 0.76) and moderate spatial ability (M = 10.74 ± 4.22).

Furthermore, the majority of the participants are novice programmers (M = 4.28 ± 0.89). The results suggest that BPM activities did not impose extraneous cognitive load on the participants. It also suggests that the participants had a moderate spatial ability to deal with any extraneous cognitive load although they lacked the expertise to work with BPM.

Table 5. Participants' cognitive load, spatial ability, and programming proficiency level.

	Cognitive load	Spatial ability	Proficiency level
Min.	1.00	3.00	4.00
Max.	7.00	18.00	13.00
Mean	2.89	10.74	4.28
Std. dev.	0.76	4.22	0.89
Skewness	1.14	1.10	1.17

4.1. Interaction patterns during BPM activities (RQ1)

The network graph (Figure 5) indicates the presence of 249 parallel edges showing the interaction between the participant nodes (numbered from P1 to P35) and non-participant nodes (denoted as Anim, Alice, Note, Teacher, Phone, Away).Analysis of the interaction patterns indicates that 36.95% of the interactions were with peers (learner–learner interaction), 34.54% were with contents (learner–content interaction), 16.87% were with distractors (learner–distractor interaction), and 11.65% were with the teacher (learner–teacher interaction). It should be noted from the graph that thick lines indicate frequently occurring interaction between the participant and non-participant nodes.

4.2. Association between interaction patterns and student factors (RQ2)

To explore the association between participants' interaction patterns during BPM activities and their gender, cognitive load, spatial ability, and programming proficiency, chi-square test of independence was conducted (see Table 6). This analysis allowed the authors to examine the relationships between these variables and identify any significant associations.

4.2.1. Association with gender

The results revealed that male students exhibited higher levels of interaction with the teacher (73%), content (65.9%), and peers (76%) compared to their female counterparts. Conversely, female students showed fewer interactions with distractors (42.3%). Although the association between gender and interaction

Table 6. Association between interaction patterns and factors.

Variables	Interaction patterns (n = number of cases)				Total
	Learner–teacher	Learner–content	Learner–learner	Learner–distractor	
Gender					
Female	248	3414	536	1658	5856
	27%	34.1%	24%	42.3%	34.3%
Male	672	6593	1696	2263	11,224
	73%	65.9%	76%	57.7%	65.7%
Total	920	10,007	2232	3921	17,080
	100%	100%	100%	100%	100%
χ^2 = 237.94, df = 3, Cramer's V = 0.118*, p-value = 0.000					
Cognitive load					
High	120	1240	144	1933	3437
	13%	12.4%	6.5%	49.3	20.1%
Moderate	200	1660	504	1052	3416
	21.7%	16.6%	22.6%	26.8%	20%
Low	600	7107	1584	936	10,227
	65.2%	71%	71%	23.9%	59.9%
Total	920	10,007	2232	3921	17,080
	100%	100%	100%	100%	100%
χ^2 = 745.80, df = 6, Cramer's V = 0.148*, p-value = 0.000					
Spatial ability					
High	384	3807	1096	672	5959
	41.7%	38%	49.1%	17.1	34.9%
Moderate	376	3696	848	1321	6241
	40.9	36.9%	38%	33.7%	36.5%
Low	160	2504	288	1928	4880
	17.4	25%	12.9%	49.2%	28.6%
Total	920	10,007	2232	3921	17,080
	100%	100%	100%	100%	100%
χ^2 = 392, df = 6, Cramer's V = 0.202*, p-value = 0.000					
Proficiency					
Expert	88	1660	72	1613	3433
	9.6%	16.6%	3.2%	41.1%	20.1%
Intermediate	64	1168	0	1200	2432
	7%	11.7%	0%	30.6%	14.2
Novice	768	7179	2160	1108	11,215
	83.5%	71.7%	96.8%	28.3%	65.7%
Total	920	10,007	2232	3921	17,080
	100%	100%	100%	100%	100%
χ^2 = 243.51, df = 6, Cramer's V = 0.367**, p-value = 0.000					

*Cramer's $V \leq 0.2$ = weak association, **Cramer's $V \leq 0.6 > 0.2$ = moderate association.

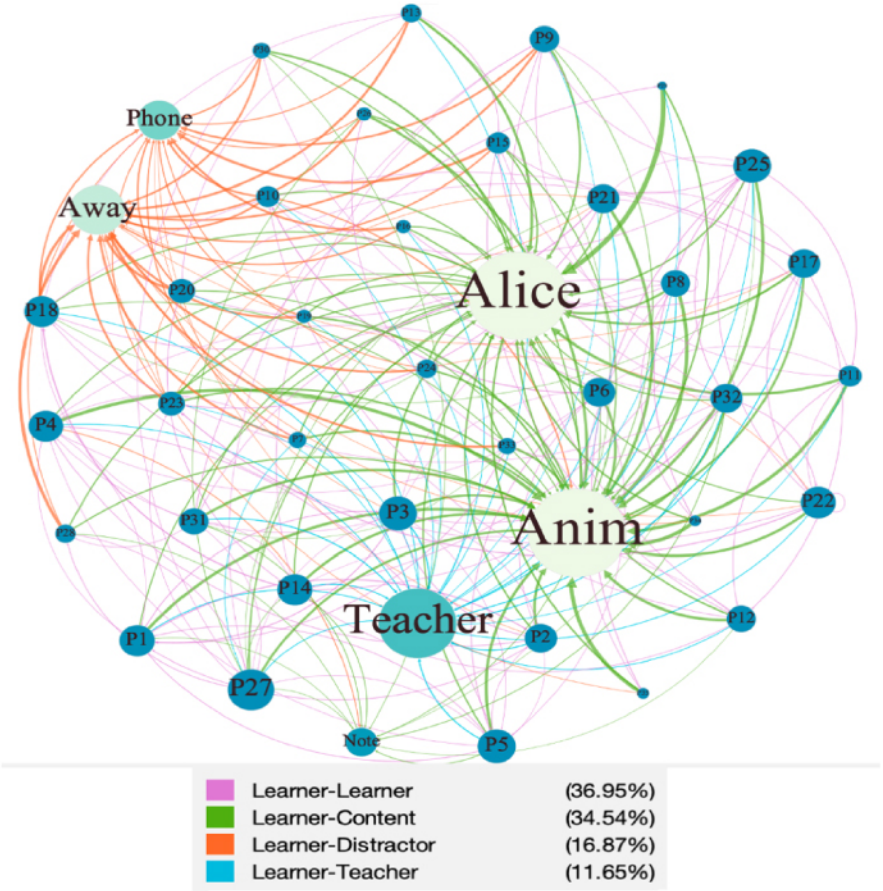

Figure 5. Interaction patterns.

patterns was significant (χ^2 = 237.94, p = 0.00), the strength of the association was weak as indicated by Cramer's V (0.118).

4.2.2. Association with cognitive load

Significant associations were observed between participants' cognitive load levels and their interaction patterns (χ^2 = 745.80, p-value = 0.00). Participants with low cognitive load tended to interact more with the teacher (65.2%), content (71%), and peers (71%) compared to those with moderate or high cognitive load levels. Conversely, participants with high cognitive load levels exhibited higher interaction with distractors (49.3%). The overall association between cognitive load and interaction patterns was weak as indicated by Cramer's V = 0.148.

4.2.3. Association with spatial ability

Significant associations were observed between participants' level of spatial ability and their interaction patterns (χ^2 = 392, p-value = 0.000). Participants with high spatial ability tended to interact more with the teacher (41.7%), content (38%), and peers (49.1%), suggesting a greater engagement with instructional materials and

collaborative learning activities. In contrast, participants with low spatial ability exhibited higher interaction with distractors (49.2%), indicating potential challenges in maintaining focus during BPM activities. The observed associations between spatial ability and interaction patterns highlight the role of cognitive factors in shaping students' engagement and participation in educational tasks.

4.2.4. Association with programming proficiency

The results also revealed significant associations between participants' programming proficiency levels and their interaction patterns (χ^2 = 243.51, p-value = 0.00). Novice programmers demonstrated higher levels of interaction with the teacher (83.5%), content (71.7%), and peers (96.8%), suggesting active engagement and learning during BPM activities. In contrast, expert programmers exhibited higher interaction with distractors (41.1%), indicating a potential tendency to explore peripheral elements rather than focusing on core instructional content. The moderate strength of the association, as indicated by Cramer's V = 0.36, suggests that programming proficiency plays a meaningful role in shaping students' interaction patterns during BPM activities.

4.3. Level of computational thinking skills during BPM activities (RQ3)

Analysis of the participants' CT scores (Table 7) shows that they obtained a mean score of 17.39 (SD = 5.60) on CT concepts and 3.48 (SD = 0.99) on CT perspectives. Out of the possible 4 points in sequences and directions, the participants obtained an average of 1.94 points (SD = 1.39). From a possible 8 points in loops, the participants obtained an average of 4.9 points (SD = 2.15). In addition, from a possible 12 points in conditionals, the participants obtained a mean score of 7.95%. A mean score of 2.59 (SD = 0.79) was also obtained in functions from a possible 4 points. From CT perspectives, the participants reported a mean perception of 3.48 (SD = 0.99): 3.33 in creative thinking, 3.88 in algorithmic thinking, 3.70 in critical thinking, 2.86 in cooperativity, and 3.49 in problem-solving. These scores suggest that the participants had moderate computational thinking skills.

4.4. Difference in students' computational thinking skills across factors (RQ4)

An independent samples t-test and analysis of variance were conducted to examine the differences in participants' CT skills across factors. In Table 8, it can be observed that there are significant differences in the CT skills across factors. An inspection of the mean scores (Table 9) shows that female students obtained higher scores in CT concepts related to sequences and directions (M = 2.25, SD = 1.47) compared to their male counterparts. The male students, on the other hand, obtained higher scores in

Table 7. Participants' computational thinking scores on concepts and perspectives.

Dimension	*N*	Mean	Std. dev.
Concept	35	17.39	5.60
Sequences	35	1.94	1.39
Loops	35	4.90	2.15
Conditionals	35	7.95	2.85
Functions	35	2.59	0.79
Perspective	35	3.48	0.99
Creative thinking	35	3.33	0.85
Algorithmic thinking	35	3.88	1.16
Critical thinking	35	3.70	1.47
Cooperativity	35	2.86	0.94
Problem-solving	35	3.49	1.02

N, number of participants assessed.

the remaining facets of CT concepts and perspectives. Concerning cognitive load, participants with low cognitive load obtained higher scores on CT concepts and perspectives with low effect sizes. On the other hand, participants with high spatial ability obtained higher scores in CT concepts and perspectives, with effect sizes ranging from high to low. Lastly, the results show a significant difference in CT scores among different programming proficiency levels. Expert programmers obtained high scores on CT concepts while novice programmers had higher perceptions on CT skills, with effect sizes ranging from high to low. These suggest that CT is a factor of gender, cognitive load, spatial ability, and programming proficiency.

4.5. Association between interaction patterns and computational thinking skills (RQ5)

To understand the factors predicting CT skills, OLR and BLR were conducted using the interaction patterns as factors (Table 10). Except for the learner–distractor interaction, all the interaction patterns significantly predict students' CT skills with significant ORs. However, learner–content and learner–learner interaction had high chances of the prediction. For example, learner–learner interaction had a higher chance to predict students' CT skills related to sequence and directions (OR = 2.20, 95% CI [3.41–3.87], p-value < 0.005), loops (OR = 6.13, 95% CI [13.11–13.87], p-value < 0.01), conditionals (OR = 5.10, 95% CI [9.15–9.78], p-value < 0.01), functions (OR = 2.16, 95% CI [1.12–3.98], p-value < 0.05), cooperativity (OR = 8.13, 95% CI [31.23–40.96], p-value < 0.01), and problem-solving (OR = 10.22, 95% CI [41.01–48.74], p-value < 0.01).

The results indicate that learner–content interaction had higher chances of predicting CT skills related to creative thinking (OR = 2.73, 95% CI [2.79–2.99],

acrt AI, Computer Science and Robotics Technology

Table 8. Significant difference in CT scores across factors.

	CT concepts				CT perspective				
Grouping	**Sequence**	**Loops**	**Conditionals**	**Functions**	**Creative**	**Algorithm**	**Critical**	**Cooperative**	**Problem-solving**
Gender	21.21*** [1.37]	5.24** [2.14]	34.07*** [2.75]	3.41** [0.78]	14.55*** [0.84]	9.36*** [1.15]	7.67*** [1.47]	17.36*** [0.93]	12.67*** [1.01]
Cognitive load	50.54*** [0.01]	517.68*** [0.05]	123.001*** [0.14]	288.81*** [0.03]	251.62*** [0.03]	322.57*** [0.04]	295.21*** [0.03]	357.78*** [0.04]	348.04*** [0.04]
Spatial ability	143.86*** [0.02]	132*** [0.14]	375*** [0.30]	371*** [0.26]	751.34*** [0.08]	494.55*** [0.06]	397.01*** [0.04]	932.30*** [0.09]	671.61*** [0.07]
Proficiency	544.42*** [0.99]	346.46*** [0.04]	574.04*** [0.06]	361.74*** [0.29]	127.82*** [0.13]	744.08*** [0.08]	561.92*** [0.06]	185.15*** [0.18]	115.63*** [0.12]

**$p < 0.05$, ***$p < 0.01$; values inside square brackets are effect sizes, values outside square brackets are the actual F or t statistics.

IntechOpen Journals

acrt AI, Computer Science and Robotics Technology

Table 9. Mean CT scores across factors.

Grouping	CT concepts				CT perspective				
	Sequence M ± SD	Loops M ± SD	Conditionals M ± SD	Functions M ± SD	Creative M ± SD	Algorithm M ± SD	Critical M ± SD	Cooperative M ± SD	Problem-solving M ± SD
Gender									
Male	1.78 ± 1.31	4.96 ± 2.16	8.47 ± 2.77	3.61 ± 0.79	4.39 ± 0.83	4.94 ± 1.10	4.76 ± 1.41	3.94 ± 0.94	4.56 ± 0.98
Female	2.25 ± 1.47	3.78 ± 2.12	6.95 ± 2.72	1.56 ± 0.76	2.19 ± 0.85	2.76 ± 1.24	2.58 ± 1.57	1.68 ± 0.91	2.35 ± 1.07
Cognitive load									
High	1.91 ± 1.37	3.07 ± 2.53	5.74 ± 3.15	1.36 ± 0.63	2.22 ± 0.93	3.49 ± 1.33	2.25 ± 1.68	1.44 ± 0.87	2.10 ± 1.14
Moderate	1.85 ± 1.35	4.27 ± 2.21	7.17 ± 2.75	2.41 ± 0.80	3.22 ± 0.93	3.63 ± 1.24	3.39 ± 1.56	2.75 ± 1.04	3.31 ± 1.12
Low	2.20 ± 1.37	6.27 ± 1.93	8.67 ± 2.47	3.69 ± 0.78	4.42 ± 0.79	4.03 ± 1.04	4.89 ± 1.35	3.97 ± 0.89	4.63 ± 0.92
Spatial ability									
High	2.90 ± 1.46	5.53 ± 1.50	9.23 ± 1.70	2.81 ± 0.65	3.58 ± 0.74	4.15 ± 0.89	4.02 ± 1.18	3.17 ± 0.88	4.78 ± 0.87
Moderate	1.75 ± 1.29	5.26 ± 1.97	8.65 ± 2.45	2.87 ± 0.67	3.37 ± 0.82	3.93 ± 1.11	3.76 ± 1.43	2.90 ± 0.91	3.54 ± 0.98
Low	1.22 ± 1.38	3.66 ± 2.49	5.50 ± 2.91	1.96 ± 0.69	2.97 ± 0.86	3.47 ± 1.34	3.24 ± 1.70	2.42 ± 0.88	2.08 ± 1.15
Proficiency									
Novice	1.01 ± 0.66	4.68 ± 1.92	7.94 ± 2.55	2.31 ± 0.65	3.53 ± 0.77	4.09 ± 0.96	3.94 ± 1.25	3.12 ± 0.90	3.72 ± 0.86
Intermediate	1.70 ± 0.01	4.85 ± 2.47	6.57 ± 3.23	2.96 ± 0.69	2.75 ± 0.80	3.24 ± 1.40	2.98 ± 1.77	2.13 ± 0.67	2.83 ± 1.15
Expert	3.00 ± 0.01	5.82 ± 2.45	9.14 ± 3.09	3.39 ± 0.61	2.99 ± 0.82	3.54 ± 1.35	3.35 ± 1.71	2.39 ± 0.80	3.11 ± 1.13

M, mean; SD, standard deviation.

Table 10. Prediction of CT skills using interaction patterns.

Interaction patterns	CT concepts				CT perspective				
	Sequence	**Loops**	**Conditionals**	**Functions**	**Creative**	**Algorithm**	**Critical**	**Cooperative**	**Problem-solving**
Learner–teacher	Est. = 1.97**	Est. = 3.03**	Est. = 6.59***	Est. = 1.67**	Est. = 1.93**	Est. = 16.53***	Est. = 15.34***	Est. = 18.86***	Est. = 15.06***
	OR = 1.66	OR = 1.03	OR = 3.91	OR = 1.32	OR = 1.77	OR = 6.93	OR = 6.40	OR = 7.61	OR = 5.96
	CI [1.78–2.16]	CI [2.82–3.24]	CI [6.27–6.92]	CL [1.42–1.91]	CI [1.86–2.01]	CI [11.85–21.22]	CL [10.67–20.01]	CI [15.25–22.48]	CI [12.42–17.71]
Learner–content	Est. = 1.28**	Est. = 8.30***	Est. = 7.96***	Est. = 1.68**	Est. = 2.89**	Est. = 48.23***	Est. = 47.61***	Est. = 18.50***	Est. = 30.50***
	OR = 1.43	OR = 5.28	OR = 4.20	OR = 1.72	OR = 2.73	OR = 10.35	OR = 10.09	OR = 7.96	OR = 8.78
	CI [1.20–1.36]	CI [8.02–8.62]	CI [7.64–8.28]	CI [1.53–1.82]	CI [2.79–2.99]	CI [41.89–54.58]	CI [41.65–53.57]	CI [14.91–22.13]	CI [26.86–34.13]
Learner–learner	Est. = 3.64***	Est. = 13.49***	Est. = 9.46***	Est. = 3.55**	Est. = 2.54**	Est. = 30.51***	Est. = 28.17***	Est. = 36.09***	Est. = 44.87***
	OR = 2.20	OR = 6.13	OR = 5.10	OR = 2.16	OR = 2.22	OR = 8.01	OR = 8.89	OR = 8.13	OR = 10.22
	CI [3.41–3.87]	CI [13.11–13.87]	CI [9.50–9.78]	CI [1.12–3.98]	CI [2.41–2.68]	CI [25.25–35.79]	CI [23.14–33.21]	CI [31.23–40.96]	CI [41.01–48.74]
Learner–distractor	Est. = −0.27*	Est. = −0.29*	Est. = −0.62*	Est. = −0.71*	Est. = −0.31*	Est. = −1.28**	Est. = −1.96**	Est. = −1.57**	Est. = −1.32**
	OR = 0.12	OR = 0.65	OR = 0.73	OR = 0.82	OR = 0.67	OR = 0.24	OR = 1.25	OR = 1.50	OR = 1.62
	CI [−1.56–1.02]	CI [−0.82–0.24]	CI [−1.66–0.43]	CI [−1.73–0.31]	CI [−1.73–1.11]	CI [−3.22–0.66]	CI [−4.16–0.25]	CI [−3.55–0.42]	CI [−3.18–0.54]

Est. = regression estimate; OR = odds ratio; CI = 95% confidence interval for lower and upper bounds; *$p > 0.05$, **$p < 0.05$, ***$p < 0$.

p-value < 0.05), algorithmic thinking (OR = 0.35, 95% CI [41.89–54.58], p-value < 0.01), and critical thinking (OR = 10.09, 95% CI [41.65–53.57]) than other interaction patterns. Although learner–distractor interaction had lower chances of predicting CT skills, such interaction posed a negative effect. Overall, it could be concluded that learner–content and learner–learner interaction are two interaction patterns during BPM activities that predict students' CT skills.

5. *Discussion*

The present study investigated students' interaction patterns during BPM activities and predicted their CT skills using the interaction patterns. Participants include 35 second-year computer science and computer education students whose classroom interactions were observed across eight interventions. The analysis revealed three important findings. First, learner–learner and learner–content interactions were the prevalent interaction patterns during BPM activities. Quite a few clusters of students engage in learner–teacher and learner–distractor patterns. From the perspective of the instructional quality model [88, 89], the prevalent incidence of learner–learner and learner–content interactions demonstrates the quality of BBPEs. The findings suggest the importance of these interaction patterns to students. At this point, the authors were compelled to believe that BBPEs have the potential to support self and collaborative programming activity. This position has been supported by considerable empirical evidence [90, 91]. However, the presence of students who often interact with distractors suggests that BPM does not always support meaningful learning. In a recent review, Yusuf and Noor [22] found that BPM is an important programming teaching tool, but it is not always effective in many experimental conditions.

The second finding revealed that interaction patterns during BPM activities differ significantly across gender, cognitive load, spatial ability, and programming proficiency. Students' CT skills also differ across these factors. With regard to gender, the study found that male students interacted more with the teacher, content, and peers compared to their female counterparts. Male students also obtained higher CT scores except for the facets of sequences and directions. Recent and previous studies have revealed that male students engage more actively in classroom activities that require critical and creative thinking as well as problem-solving skills [92] while significant differences exist in favor of females for verbal fluency [93]. Nevertheless, there is no widely accepted empirical evidence indicating that interaction with BBPEs favors a particular gender category. Several studies have supported these findings (e.g., [28, 29, 94–96]), yet several others have reported no difference [74, 75, 97]. Concerning gender differences in CT skills, results are also mixed across the literature. For example, Niousha *et al.* [72] found a gender difference in CT skills in favor of boys while Sun *et al.* [11] found a significant difference in favor of girls. Maintaining a neutral stand, Espino and

Gonzalez [94] acknowledge the existence of gender differences in computing activities but argued that everyone is capable of developing substantial CT skills regardless of their gender category. Gender neutrality in CT skills was also reported by Oluk and Korkmaz [74] and Sun *et al.* [98].

A plausible explanation for the gender difference is rooted in gender theory [99]. This theory conceptualizes gender as categories of social expectations, roles, and behaviors. Although this view remains controversial in the literature, it does infer that some social activities are gender-stereotyped. Proponents of this view argue that disciplines such as mathematics and computing are perceived to be masculine and, therefore, females are more likely to struggle in these disciplines. In their previous study, Espino and Gonzalez [94] raised the issue of stereotypical gender preferences in the context of BPM and argued that the reported gender differences in computing and CT skills might be due to the compatibility of certain computing activities to a specific gender category. They further explained that males generally preferred programming constructs, which are reflected more in their CT scores.

However, as science educators who always understand and appreciate the need for gender inclusivity, the authors maintain a neutral stand on these justifications but believe that such differences might partly manifest from the lens of geographical context. For example, in Nigeria, enrollment into computing courses is generally skewed in favor of boys (85.87%), thereby creating a gender disparity [100]. While the authors believe that BPM is for everyone, it is worth noting that girls lagged far behind boys in computing courses in most African countries including Nigeria. For this reason, expanding programming tools to meet the expectations of the girl child remains the authors' top priority.

For cognitive load, the study found that participants with low cognitive load interacted more with the teacher, contents, and peers compared to their counterparts with high and moderate cognitive load. These differences also hold for CT skills. The effect of cognitive load when working with BBPEs has been widely reported. In a recent review, Yusuf and Noor [22] reported the pedagogical effectiveness of BBPEs in promoting CT skills but highlighted its cognitive load effects. The authors reported that learners often struggle to interact with these tools due to the transient attribute associated with their virtual realities. Other studies also reported this problem when students are exposed to coding using BBPEs [51, 66].

Although cognitive load was found to affect CT skills and meaningful interaction in the BPM classroom, this effect is counterbalanced by students' spatial ability. The study found evidence of the enhancer hypothesis [68], which assumes that high spatial ability learners benefit more from dynamic visualization compared to low spatial ability learners. Prior studies also reported evidence of the enhancer hypothesis in the context of BPM [67, 70]. Literature examining the effect of

cognitive load and spatial ability on CT skills and BPM activities is scant. Few available studies found that CT is a factor of spatial ability and cognitive load [51, 66], thus supporting the findings.

The study also found some evidence of the expertise reversal effect because novice programmers significantly benefited from the BPM activities as indicated by their meaningful interaction than the intermediates and experts. Novice programmers were also reported to have higher scores on CT perspectives while expert programmers obtained high scores on CT concepts. The evidence of expertise reversal effect in this study is an indication that most BBPEs are not inclusive; they are largely beneficial to a typical expert category but become counterproductive to others and vice versa. In support of this finding, Spanjers *et al.* [101] found that these programming environments were more efficient for novice learners but not for students with higher knowledge proficiency.

Explanations have been offered for the reason of the expertise reversal effect in most block-based environments. Kalyuga [27] explains that when information presented in these environments is familiar to the learners, they easily process its transience and ignore the content because of their potential and anticipation for higher mental objects. Prior research has also shown that measures to improve students' learning outcomes using block-based environments as additional instructional guidance are often more beneficial to novices and counterproductive for expert learners, who do not need additional instructional guidance [102]. The experts have to reconcile their guidance in their schema with the additional guidance, which might further induce extraneous load.

It is quite surprising that expert programmers performed better in CT concepts than novice programmers despite having low meaningful interaction during the BPM activities. This difference is also explained in the context of expertise reversal effect. For instance, Aysolmaz and Reijers [26] explain that expertise reversal effect also occurs with a high level of expertise. In this situation, highly interactive programming environments, including those that were designed to enhance deep exploration of CT concepts, may overload the memory of novice learners and benefit the expert learners. This argument confirms the validity of the present finding, which indicates that participants with different types of expertise benefited more from the programming environment depending on the concerned CT dimension.

The third finding revealed that interaction patterns during BPM activities significantly predict CT skills. Specifically, two interaction patterns were found to significantly predict CT. Learner–learner interaction had a higher chance of predicting students' CT skills related to sequence and directions, loops, conditionals, functions, cooperativity, and problem-solving skills. On the other hand, learner–content interaction had higher chances of predicting CT skills related to

creative thinking, algorithmic thinking, and critical thinking. Although learner–distractor interaction had lower chances of predicting CT skills, such interaction posed a negative effect. There is a consensus in the literature claiming that students' skills in BBPEs significantly correlate with their CT skills [5, 6, 74]. A practical implication of this finding points to the need for active participation of students in BPM activities. The finding also suggests the need for collaborative activities along with the integration of interactive animation for effective learning of programming and improvement of CT skills. To this end, although the acquisition of CT skills may differ across factors, engaging students in collaborative programming activities using interactive BPM would provide an enabling environment for students to acquire more CT skills.

Overall, the above findings suggest the crucial role of Alice in this research. Firstly, it provides a user-friendly interface that lowers the barriers for students new to programming. Its drag-and-drop functionality and visual representation of code enable learners to grasp fundamental programming concepts more easily, fostering engagement and participation. Secondly, by immersing students in programming tasks within the Alice environment, the authors were able to observe and analyze their interaction patterns comprehensively. From how they navigate instructional materials to how they collaborate with peers, Alice serves as the context for capturing these behaviors. This allows the authors to gain insights into the dynamics of student engagement and the impact of different interaction modalities on learning outcomes. Moreover, Alice facilitates the integration of computational thinking assessment into the study. Through the BPM activities, we assessed students' CT skills along with factors contributing to these skills.

6. Conclusion

This study has shown the possibilities of classifying students' interaction patterns using data obtained from time-dependent distribution. These interaction patterns are of significant importance because they appear to predict meaningful learning. By employing statistical models, the study found that students' CT skills and their interaction with BBPEs are factors of gender, cognitive load, spatial ability, and programming proficiency. Despite these factors, interaction patterns during BPM activities predict students' CT skills. The research has provided opportunities for educators to integrate BBPEs in learning programming and CT concepts. Although such integration is likely to occur with the help of strong educational policies, teachers are encouraged to cultivate the spirit of collaboration in students as collaborative activities in this research were found to predict CT skills more than other interaction patterns. To advance research on CT and BPM, it is imperative to always consider learners' demographic profiles as they play important roles in meaningful learning. More importantly, students' gender should be given more

consideration due to its sensitivity to computing research particularly in sub-Saharan Africa, where girl-child education in computing is overly neglected.

While acknowledging the validity of the study findings, various limitations could affect such validity. First is the limited sample size. Although similar interaction patterns are expected in other studies that employ larger samples, more valid findings are predicted due to more diversity. However, larger samples could also pose difficulty in coding larger amounts of video data. Second, the video data was collected by a researcher-coded approach, which is prone to errors due to the strenuous task of coding a large amount of information. Although the researchers believe in the potential of technology to automatically identify interaction features using computer vision and deep learning algorithms, such an approach is expensive and could also lead to coding errors because some false faces might be captured as true faces, especially when there are blurred images. Despite these limitations, the study still retains substantial validity.

Conflict of interest

The authors declare no conflict of interest.

Source data

Source data (raw scientific data accompanying the research) for this article is available on Figshare: https://doi.org/10.5772/acrt.deposit.26317015

References

1 Yusuf A, Noor NM. Revising the computer programming attitude scale in the context of attitude ambivalence. *J Comput Assist Learn*. 2023a;**39**(6):1751–1768. doi:10.1111/jcal.12838.

2 Sun L, Liu J. Different programming approaches on primary students' computational thinking: a multifactorial chain mediation effect. *Educ Technol Res Develop*. 2023;**72**: 557–584. doi:10.1007/s11423-023-10312-2.

3 Bau D, Gray J, Kelleher C, Sheldon J and Turbak F. Learnable programming: Blocks and beyond. *Commun ACM*. 2017;**60**: 72–80. doi: 10.1145/3015455.

4 Weintrop D and Wilensky U. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Trans Comput Edu*. 2017;**18**(1):1–25. doi: 10.1145/3089799.

5 Angeli C. The effects of scaffolded programming scripts on pre-service teachers' computational thinking: developing algorithmic thinking through programming robots. *Int J Child-Comput Interact*. 2022;**31**: 100329. doi:10.1016/j.ijcci.2021.100329.

6 Fagerlund J, Hakkinen P, Vesisenaho M, Viiri J. Computational thinking in programming with Scratch in primary schools: a systematic review. *Comput Appl Eng Educ*. 2021;**29**(1):12–28. doi:10.1002/cae.22255.

7 Sun D, Ouyang F, Li Y, Zhu C. Comparing learners' knowledge, behaviors, and attitudes between two instructional modes of computer programming in secondary education. *Int J STEM Educ*. 2021;**8**: 54. doi:10.1186/s40594-021-00311-1.

8 Tikva C, Tambouris EA. Systematic mapping study on teaching and learning computational thinking through programming in higher education. *Think Ski Creat*. 2021;**41**: 100849. doi:10.1016/j.tsc.2021.100849.

9 Yang W, Kit Ng DT, Su J. The impact of story-inspired programming on preschool children's computational thinking: a multi-group experiment. *Think Ski Creat*. 2023;**47**: 101218. doi:10.1016/j.tsc.2022.101218.

10 Popat S, Starkey L. Learning to code or coding to learn? A systematic review. *Comput Educ*. 2019;**128**: 365–376. doi:10.1016/j.compedu.2018.10.005.

11 Sun L, Hu L, Zhou D. Programming attitudes predict computational thinking: analysis of differences in gender and programming experience. *Comput Educ*. 2022;**181**: 104457. doi:10.1016/j.compedu.2022.104457.

12 Rozali NF, Zaid NM, Noor NM, Ibrahim NH. Developing a unified model of teaching computational thinking. In: *2018 IEEE 10th International Conference on Engineering Education (ICEED), Kuala Lumpur, Malaysia*. Malaysia: IEEE; 2018. p. 208–213. doi:10.1109/ICEED.2018.8626930.

13 Tedre M, Denning P. The long quest for computational thinking. In: *16th Koli Calling International Conference on Computing Education Research*. New York, NY: ACM; 2016. p. 120–129. doi:10.1145/2999541.2999542.

14 Wing JM. Computational thinking. *Commun ACM*. 2006;**49**(3):33–35. doi:10.1145/1118178.1118215.

15 Cheng Y-P, Lai C-F, Chen Y-T, Wang W-S, Huang Y-M, Wu T-T. Enhancing student's computational thinking skills with student-generated questions strategy in a game-based learning platform. *Comput Educ*. 2023;**200**: 1–20. doi:10.1016/j.compedu.2023.104794.

16 Wong GKW. Amplifying children's computational problem-solving skills: a hybrid-based design for programming education. *Educ Inf Technol*. 2024;**29**: 1761–1793. doi:10.1007/s10639-023-11880-9.

17 Kong SC. Components and methods of evaluating computational thinking for fostering creative problem-solvers in senior primary school education. In: Kong SC and Abelson H., editors. *Computational thinking education*. Singapore: Springer; 2019. p. 187–204. https://doi.org/10.1007/978-981-13-6528-7_8.

18 Resnick M, Maloney J, Monroy-Hernández A, Rusk N, Eastmond E, Brennan K, et al. Scratch: programming for all. *Commun ACM*. 2009;**52**: 60–67. doi:10.1145/1592761.1592779.

19 Pausch R, Burnette T, Capeheart AC, Conway M, Cosgrove D, Deline R, et al. Alice: rapid prototyping system for virtual reality. *IEEE Comput Graph Appl*. 1995;**15**: 8–11. doi:10.1109/38.376600.

20 Biju SM. Taking advantage of Alice to teach programming concepts. *E-Learning Digital Media*. 2013;**10**: 22–29. doi:10.2304/elea.2013.10.1.22.

21 Dwarika J, Ruth de Villiers MR. Use of the Alice visual environment in teaching and learning object-oriented programming. In: *Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists*. New York, USA: Association for Computing Machinery; 2015. p. 1–14. Article 14. doi:10.1145/2815782.2815815.

22 Yusuf A, Noor NM. Research trend on learning computer programming with program animation: a systematic mapping study. *Comput Appl Eng Educ*. 2023b;**31**(6):1552–1582. doi:10.1002/cae.22659.

23 Sweller J, Ayres P, Kalyuga S. *Cognitive load theory*. New York: Springer; 2011.

24 Awasekar DD. Effect of program visualization to teach computer programming in a resource constrained classroom. In: *IEEE Fifth International Conference on Technology for Education*. Kharagpur: IEEE; 2013. p. 93–100. doi:10.1109/T4E.2013.31.

25 Meerbaum-Salant O, Armoni M, Ben-Ari M. Learning computer science concepts with Scratch. *Comput Sci Educ*. 2013;**23**: 239–264. doi:10.1080/08993408.2013.832022.

26 Aysolmaz B, Reijers HA. Animation as a dynamic visualization technique for improving process model comprehension. *Inf Manag*. 2021;**58**(5):103478. doi:10.1016/j.im.2021.103478.

27 Kalyuga S. Relative effectiveness of animated and static diagrams: an effect of learner prior knowledge.

Comput Human Behav. 2008;**24**(3):852–861. doi:10.1016/j.chb.2007.02.018.

28 Montuori C, Ronconi L, Vardanega T, Arfe B. Exploring gender differences in coding at the beginning of primary school. *Front Psychol*. 2022;**13**: 887280. doi:10.3389/fpsyg.2022.887280.

29 Roman-Gonzalez M, Perez-Gonzalez J-C, Jimenez-Fernandez C. Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Comput Human Behav*. 2017;**72**: 678–691. doi:10.1016/j.chb.2016.08.047.

30 Price CB, Price-Mohr R. Exploring gender differences in primary school computer programming classes: a study in an English state-funded urban school. *Int J Primary, Elementary Early Years Educ*. 2023;**51**(2):306–319. doi:10.1080/03004279.2021.1971274.

31 Bers MU, Flennery L, Kazakoff ER, Sullivan A. Computational thinking and tinkering: exploration of an early childhood robotics curriculum. *Comput Educ*. 2014;**72**: 145–157. doi:10.1016/j.compedu.2013.10.020.

32 Namli NA, Aybek B. An investigation of the effect of block-based programming and unplugged coding activities on fifth graders' computational thinking skills, self-efficacy and academic performance. *Contemp Educ Technol*. 2022;**14**(1):ep341. doi:10.30935/cedtech/11477.

33 Ou Yang F-C, Lai H-M, Wang Y-W. Effect of augmented reality-based virtual educational robotics on programming students' enjoyment of learning, computational thinking skills, and academic achievement. *Comput Educ*. 2023;**195**: 1–22. doi:10.1016/j.compedu.2022.104721.

34 Sun D, Ouyang F, Li Y and Zhu C. Comparing learners' knowledge, behaviors, and attitudes between two instructional modes of computer programming in secondary education. *Int J STEM Edu*. 2021;**8**: 54. doi:10.1186/s40594-021-00311-1.

35 Agbo FJ, Okpanachi LO, Ocheja P, Oyelere SS, Sani G. How can unplugged approach facilitate novice students' understanding of computational thinking? An exploratory study from a Nigerian university. *Think Ski Creat*. 2024;**51**: 101458. doi:10.1016/j.tsc.2023.101458.

36 Wing JM. Research notebook: computational thinking – what and why? The link. The magazine of the Carnegie Mellon University School of Computer Science [Internet]; 2011. Available from: http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

37 Aho AV. Computation and computational thinking. *Comput J*. 2012;**55**(7):832–835. doi:10.1093/comjnl/bxs074.

38 Yadav A, Hong H, Stephenson C. Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *Tech Trends*. 2016;**60**: 565–568. doi:10.1007/s11528-016-0087-7.

39 Fagerlund J, Hakkinen P, Vesisenaho M and Viiri J. Computational thinking in programming with Scratch in primary schools: A systematic review. *Comput Appl Eng Edu*. 2021;**29**(1):12–28. doi:10.1002/cae.22255.

40 Brennan K, Resnick M. New frameworks for studying and assessing the development of computational thinking. In: *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*, vol. 1, Canada: Scientific Research Publishing; 2012. 25 p.

41 Repenning A, Basawapatna A, Escherle N. Computational thinking tools. In: *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Cambridge, UK*. Cambridge: IEEE; 2016. p. 218–222. doi:10.1109/VLHCC.2016.7739688.

42 Shute VJ, Sun C, Asbell-Clarke J. Demystifying computational thinking. *Educ Res Rev*. 2017;**22**: 142–158. doi:10.1016/j.edurev.2017.09.003.

43 Korkmaz O, Cakir R, Ozden MY. A validity and reliability study of the computational thinking scales (CTS). *Comput Human Behav*. 2017;**72**: 558–569. doi:10.1016/j.chb.2017.01.005.

44 Doleck T, Bazelais P, Lemay D, Saxena A, Basnet R. Algorithmic thinking, cooperativity, creativity, critical thinking, problem solving: exploring the relationship between computational thinking and academic performance. *J Comput Educ*. 2017;**4**(4):355–369. doi:10.1007/s40692-017-0090-9.

45 Çakiroğlu Ü, Çevik İ. A framework for measuring abstraction as a sub-skill of computational thinking in block-based programming environments. *Educ Technol Soc*. 2022;**27**: 9455–9484. doi:10.1007/s10639-022-11019-2.

46 Werner L, Denner J, Campe S, Kawamoto DC. The fairy performance assessment: measuring computational thinking in middle school. In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. Raleigh, NC: ACM; 2012. p. 215–220. doi:10.1145/2157136.2157200.

47 Chen G, Shen J, Barth-Cohen L, Jiang S, Huang X, Eltoukhy M. Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Comput Educ*. 2017;**109**: 162–175.

48 Guggemos J, Seufert S, Román-González M. Computational thinking assessment – towards more vivid interpretations. *Technol Knowl Learn*. 2023;**28**: 539–568. doi:10.1007/s10758-021-09587-2.

49 Roman-Gonzalez M. Computational thinking test: design guidelines and content validation. In: *Proceedings of EDULEARN15 Conference 6th–8th July 2015, Barcelona, Spain*. IATED; 2015. Available from: https://library.iated.org/view/ROMANGONZALEZ2015COM.

50 Moon H, Cheon J, Kwon K. Difficult concepts and practices of computational thinking using block-based programming. *Int J Comput Sci Educ Schools*. 2022;**5**(3):3–16. doi:10.21585/ijcses.v5i3.129.

51 Chen X, Wang X. Computational thinking training and deep learning evaluation model construction based on Scratch modular programming course. *Comput Intell Neurosci*. 2023;**2023**: 3760957. doi:10.1155/2023/3760957.

52 Guggemos J, Seufert S and Román-González M. Computational Thinking Assessment – Towards more vivid interpretations. *Technol Knowled Learning*. 2023;**28**:539–568. doi: 10.1007/s10758-021-09587-2.

53 Román-González M., Moreno-León J and Robles G. Combining assessment tools for a comprehensive evaluation of Computational Thinking interventions. In: Kong S-C and Abelson H., editors. *Computational thinking education*. Springer; 2019. p. 79–98.

54 Chan S-W, Looi C-K, Sumintono B. Assessing computational thinking abilities among Singapore secondary students: a Rasch model measurement analysis. *J Comput Educ*. 2021;**8**: 213–236. doi:10.1007/s40692-020-00177-2.

55 Hsu T-C, Chang C, Lin Y-W. Effects of voice assistant creation using different learning approaches on performance of computational thinking. *Comput Educ*. 2023;**192**: 104657. doi:10.1016/j.compedu.2022.104657.

56 Kong S-C, Lai M. Effects of a teacher development program on teachers' knowledge and collaborative engagement, and students' achievement in computational thinking concepts. *Br J Educ Technol*. 2023;**54**(4):489–512. doi:10.1111/bjet.13256.

57 Wang X, Cheng M and Li X. Teaching and learning computational thinking through game-based learning: A systematic review. *J Educ Comput Res*. 2023;**61**(7):1505–1536. doi: 10.1177/07356331231180951.

58 Sentance S and Csizmadia A. Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*. 2017;**22**(2):469–495. doi: 10.1007/s10639-016-9482-0.

59 Czerkawski BC and Lyman EW. Exploring issues about computational thinking in higher education.*Tech Trends*. 2015;**59**(2):57–65. doi: 10.1007/s11528-015-0840-3.

60 Corral L, Fronza I, Pahl C. Block-based programming enabling students to gain and transfer knowledge with a no-code approach. In: *Proceedings of the 22st Annual Conference on Information Technology Education (SIGITE 21)*. NY: Association for Computing Machinery; 2021. p. 55–56. doi:10.1145/3450329.3478314.

61 Kumar P, Saxena C, Baber H. Learner–content interaction in e-learning – The moderating role of perceived harm of COVID-19 in assessing the satisfaction of learners. *Smart Learn Environ*. 2021;**8**: 5. doi:10.1186/s40561-021-00149-8.

62 Moore MG. Editorial: three types of interaction. *Am J Distance Educ*. 1989;**3**(2):1–7. doi:10.1080/08923648909526659.

63 Goldberg P, Wanger W, Seidel T, Sturmer K. Why do students exhibit different attention-related behavior during instruction? Investigating effects of individual and context-dependent determinants. *Learn Instruct*. 2023;**83**: 1016694. doi:10.1016/j.learninstruc.2022.101694.

64 Hopcan S, Polat E, Albayrak E. Collaborative behavior patterns of students in programming instruction. *J Educ Comput Res*. 2022;**60**(4):1035–1062. doi:10.1177/07356331211062260.

65 Olsson J, Granberg C. Teacher–student interaction supporting students' creative mathematical reasoning during problem-solving using Scratch. *Math Think Learn*. 2022;**26**(3):278–305. doi:10.1080/10986065.2022.2105567.

66 Pellas N. Exploring relationships among students' computational thinking skills, emotions, and cognitive load using simulation games in primary education. *J Comput Assist Learn*. 2023;**39**(5):1576–1590. doi:10.1111/jcal.12819.

67 Chikha AB, Khacharem A, Trabelsi K, Bragazzi NL. The effect of spatial ability in learning from static and dynamic visualizations: a moderation analysis in 6-year-old children. *Front Psychol*. 2021;**12**: 583968. doi:10.3389/fpsyg.2021.583968.

68 Höffler TN. Spatial ability: its influence on learning with visualizations – a meta-analytic review. *Educ Psychol Rev*. 2010;**22**: 245–269. doi:10.1007/s10648-010-9126-7.

69 Höffler TN, Leutner D. The role of spatial ability in learning from instructional animations – evidence for an ability-as-compensator hypothesis. *Comput Human Behav*. 2011;**27**(1):209–216. doi:10.1016/j.chb.2010.07.042.

70 Huk T. Who benefits from learning with 3D models? The case of spatial ability. *J Comput Assist Learn*. 2006;**22**(6):392–404. doi:10.1111/j.1365-2729.2006.00180.x.

71 Al-Linjawi AA, Al-Nuaim HA. Using Alice to teach novice programmers OOP concepts. *J King Abdulaziz Univ*. 2010;**22**: 59–68. doi:10.4197/Sci.22-1.4.

72 Niousha R, Saito D, Washizaki H, Fukazawa Y. Investigating the effect of binary gender preferences on computational thinking skills. *Educ Sci*. 2023;**13**: 433. doi:10.3390/educsci13050433.

73 Statter D and Armoni M. Learning abstraction in computer science: A gender perspective. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE '17)*; 2017. p. 5–14. doi: 10.1145/3137065.3137081.

74 Oluk A, Korkmaz O. Comparing students' scratch skills with their computational thinking skills in terms of different variables. *Int J Modern Educ Comput Sci*. 2016;**8**(11):1–7. doi:10.5815/ijmecs.2016.11.01.

75 Price CB and Price-Mohr R. Exploring gender differences in primary school computer programming classes: A study in an English state-funded urban school. *Int J Primary Element Early Years Edu*. 2021;**51**(2):306–319. doi: 10.1080/03004279.2021.1971274.

76 Andrade A, Delandshere G, Danish JA. Using multimodal learning analytics to model student behaviour: a systematic analysis of behavioural framing. *J Learn Anal*. 2016;**3**(2):282–306. doi:10.18608/jla.2016.32.14.

77 Yusuf A, Noor NM, Bello S. Using learning analytics to model students' learning behavior in animated programming classroom. *Educ Inf Technol*. 2023;**29**: 6947–6990. doi:10.1007/s10639-023-12079-8.

78 Creswell JW. *Research design: qualitative, quantitative and mixed method approaches*. 4th ed. Thousand Oaks: Sage Publications; 2014.

79 Molina-Ayuso A, Adamuz-Povedano N, Bracho-Lopez R, Torralbo-Rodriguez M. Introduction to computational thinking with Scratch for teacher training for Spanish primary school teachers in Mathematics. *Educ Sci*. 2022;**12**: 899. doi:10.3390/educsci12120899.

80 Kalyuga S, Chandler P, Sweller J. Managing split-attention and redundancy in multimedia instruction. *Appl Cogn Psychol*. 1999;**13**: 351–371. doi:10.1002/(SICI)1099-0720(199908)13:4<351::AID-ACP589>3.0.CO;2-6.

81 Paas F. Training strategies for attaining transfer of problem-solving skill in statistics: a cognitive load approach. *J Educ Psychol*. 1992;**84**: 429–434. doi:10.1037/0022-0663.84.4.429.

82 Paas F, Tuovinen J, Tabbers HK, Van Gerven PWM. Cognitive load measurement as a means to advance cognitive load theory. *Educ Psychol*. 2003;**38**: 63–71. doi:10.1207/S15326985EP3801_8.

83 Ekstrom RB, French JW, Harmann HH. *Manual for kit of factor-referenced cognitive tests*. Princeton, NJ: Educational Testing Service; 1976.

84 Kesselbacher M, Bollin A. Quantifying patterns and programming strategies in block-based programming environments. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada*. Montreal: IEEE; 2019. p. 254–255. doi:10.1109/ICSE-Companion.2019.00101.

85 Obermuller F, Pernerstorfer R, Bailey L, Heuer U, Fraser G. Common patterns in block-based robot programs. In: *Proceedings of the 17th Workshop in Primary and Secondary Computing Education*. New York: Association for Computing Machinery; 2022. p. 1–10. Article No. 4. doi:10.1145/3556787.3556859.

86 Cohen J. A coefficient of agreement for nominal scales. *Educ Psychol Meas*. 1960;**20**: 37–46. doi:10.1177/001316446002000104.

87 Chen X, Feng S. Exploring the relationships between social presence and teaching presence in online video-based learning. *J Comput Assist Learn*. 2023;**2023**: 3760957. doi:10.1111/jcal.12843.

88 Praetorius A-K, Klieme E, Herbert B, Pinger P. Generic dimensions of teaching quality: the German framework of three basic dimensions. *ZDM Math Educ*. 2018;**50**(3):407–426. doi:10.1007/s11858-018-0918-4.

89 Scherer R, Nilsen T, Jansen M. Evaluating individual students' perceptions of instructional quality: an investigation of their factor structure, measurement invariance, and relations to educational outcomes. *Front Psychol*. 2016;**7**: 110. doi:10.3389/fpsyg.2016.00110.

90 Iskrenovic-Momcilovic O. Pair programming with scratch. *Educ Inform Technol*. 2019;**24**: 2943–2952. doi:10.1007/s10639-019-09905-3.

91 Andersen R, Mørch AI, Torine Litherland K. Collaborative learning with block-based programming: investigating human-centered artificial intelligence in education. *Behav Inf Technol*. 2022;**41**(9):1830–1847. doi:10.1080/0144929X.2022.2083981.

92 Shubina I, Kulakli A. Critical thinking, creativity and gender differences for knowledge generation in education. *Read Writ Q*. 2019;**10**(1):3086–3093. doi:10.20533/licej.2040.2589.2019.0405.

93 Matud P, Rodriguez C, Grande J. Gender differences in creative thinking. *Pers Individ Differ*. 2007;**43**(5):1137–1147. doi:10.1016/j.paid.2007.03.006.

94 Espino EE, Gonzalez CG. Gender and computational thinking: review of the literature and applications. In: *Proceedings of the XVII International Conference on Human–Computer Interaction*. New York: Association for Computing Machinery; 2016. p. 1–2. Article No. 46. doi:10.1145/2998626.2998665.

95 Jiang B, Li Z. Effect of scratch on computational thinking skills of Chinese primary school students. *J Comput Educ*. 2021;**8**: 505–525. doi:10.1007/s40692-021-00190-z.

96 Yucel Y, Rizvanoglu K. Battling gender stereotypes: a user study of a code-learning game, "Code Combat," with middle school children. *Comput Human Behav*. 2019;**99**: 352–365. doi:10.1016/j.chb.2019.05.029.

97 Kožuh I, Krajnc R, Hadjileontiadis LJ, Debevc M. Assessment of problem-solving ability in novice programmers. *PLoS ONE*. 2018;**13**: e0201919. doi:10.1371/journal.pone.0201919.

98 Sun L, Hu L, Zhou D. Improving 7th-graders' computational thinking skills through unplugged programming activities: a study on the influence of multiple factors. *Think Ski Creat*. 2021;**42**: 100926. doi:10.1016/j.tsc.2021.100926.

99 Jule A. Gender theory. In: Michalos AC, editor. *Encyclopedia of quality of life and well-being research*. Dordrecht: Springer; 2014. p. 2464–2466. doi:10.1007/978-94-007-0753-5_1137.

100 Statista. Number of undergraduate students at universities in Nigeria as of 2019, by gender and discipline [Internet]; 2021. Available from: https://www.statista.com/statistics/1262928/number-of-undergraduate-students-at-universities-in-nigeria-by-gender-and-discipline/.

101 Spanjers IAE, van Gog T and van Merriënboer JJG. Segmentation of worked examples: Effects on cognitive load and learning. *Appl Cognitive Psychol*. 2011;**26**(3):353–358. doi: 10.1002/acp.1832

102 Kalyuga S, Ayres P, Chandler P, Sweller J. The expertise reversal effect. *Educ Psychol*. 2003;**38**(1):23–31. doi:10.1207/S15326985EP3801_4.