Education and Information Technologies (2023) 28:6845–6867
https://doi.org/10.1007/s10639-022-11465-y

The effect of scaffolding programming games and attitudes towards programming on the development of Computational Thinking

Christina Tikva[1] · **Efthimios Tambouris**[1]

Received: 18 July 2022 / Accepted: 9 November 2022 / Published online: 23 November 2022

Abstract
Teaching and learning Computational Thinking (CT) is at the forefront of educational interest. In the process of teaching and learning CT, learning strategies and tools play an important role. Efforts have been made to apply several learning strategies for teaching Computational Thinking. Among them, game-based learning and scaffolding are widely adopted. However, more research is needed on how the absence and presence of scaffolding strategies in programming games could affect students' cognitive CT learning gains. This study aims to investigate the effect of scaffolding programming games on the development of middle school students' CT. In addition, herein we aim to explore the effect of students' programming attitudes in their CT development. To this end, students were introduced to CT under two distinct experimental conditions: a scaffolding version of a programming game and a non-scaffolding version of the same game. Results report statistically significant differences between the pre- and post-intervention CT scores for all students and statistically significant improvement in learning outcomes in favor of the scaffolding group. In addition, the study hypothesized that attitudes towards programming would have an impact on students' CT. Although this hypothesis has not been confirmed, the results suggest that students who have a less positive attitude towards programming could particularly benefit from scaffolding aspects in programming games.

Keywords Computational Thinking · Scaffolding · Programming Games · Programming attitudes

✉ Christina Tikva
ch.tikva@uom.edu.gr

[1] University of Macedonia, 156 Egnatia Street, GR-546 36 Thessaloniki, Greece

1 Introduction

Teaching and learning Computational Thinking is at the forefront of educational interest. Wing (2008) argues that Computational Thinking involves formulating problems and their solutions, so that solutions are represented in a form that can be effectively carried out by an information processing agent, considering Computational Thinking as an essential skill for everyone. Many other educators and researchers support that Computational Thinking is a fundamental skill for twenty-first century students of all ages. Furthermore, particular interest has been given to the integration of programming into K-12 education as means of developing Computational Thinking (García-Peñalvo & Mendes, 2018; Kong et al., 2020). According to the meta-review by Hsu et al. (2018), Computational Thinking through Programming focuses primarily on elementary and middle school students, with many countries updating their curricula to integrate programming into K-12 education (Heintz et al., 2016).

In the process of teaching and learning Computational Thinking, learning strategies play an important role. Efforts have been made to investigate several pedagogies and learning strategies for teaching Computational Thinking. Among them, game-based learning and scaffolding are widely adopted (Hsu et al., 2018). Game-based approaches can increase student motivation, address their disengagement, and foster the acquisition of Computational Thinking (Weintrop et al., 2016). Thus, they are exploited in several studies (e.g., de Souza et al., 2019; Garneli & Chorianopoulos, 2018, 2019; Israel-Fishelson & Hershkovitz, 2020; Zhao & Shute, 2019). In addition to game-based learning, scaffolding is proposed (Repenning et al., 2015) to increase motivation and student participation in Computational Thinking. Studies also (e.g., Angeli & Valanides, 2020) reveal that there is a need to scaffold students' learning during their engagement with Computational Thinking. According to Denner et al. (2012), without proper guidance students face significant challenges in developing Computational Thinking skills. Scaffolding helps students better understand Computational Thinking concepts, which they would not be able to assimilate if left alone to experiment in a programming environment (Grover et al., 2015). The aforementioned efforts highlight the importance of feedback and guidance strategies in Computational Thinking approaches. However, more research is needed on how the absence versus presence of scaffolding strategies could affect students' cognitive Computational Thinking learning gains.

Technologies and tools are also important. Thus, researchers focus on the development of tools specific to support Computational Thinking learning through programming. Sengupta et al. (2013) developed the CTSiM (Computational Thinking in Simulation and Modeling) tool. CTSiM is a visual programming environment that includes a modeling environment and supports low-threshold, high-ceiling, algorithm visualization, scaffolding and constructivist learning activities. The second version of CTSiM is developed to provide students with adaptive scaffolding based on modeling learner's domain knowledge, cognitive skills and interests (Basu et al., 2017). Weintrop et al. (2016) developed a constructionist video game aiming to foster Computational Thinking. RobotBuilder features a block-based programming

language to allow students to construct their game strategies. Clark and Sengupta (2019) developed the SURGE: Gameblox, a Disciplinary-Integrated Game (DIG). SURGE: Gameblox exploits formal representations (such as scientific graphs) and agent-based game programming in a collaborative environment targeting on promoting Computational Thinking. Although the aforementioned tools have been developed to include features that support specific learning strategies, more empirical research that aims to investigate the relationship between tools, learning strategies and Computational Thinking development (Tikva & Tambouris, 2021b) is needed.

In addition to learning strategies and tools, research studies are interested in how various factors influence the acquisition of Computational Thinking. Research (e.g., Kong et al., 2018) has focused on exploring students' attitudes towards programming in the context of Computational Thinking. Particular interest has been paid on how several Computational Thinking interventions could improve students' attitudes towards programming. For example, Cetin (2016) explored the effect of a Scratch-based intervention on students' attitudes towards programming. However, studies that explore the relationship between attitudes towards programming and Computational Thinking acquisition are scarce (Sun et al., 2022).

Therefore, the present study aims to investigate the effect of scaffolding programming games on middle school students' Computational Thinking acquisition. An additional goal is to explore the effect of middle school students' attitudes towards programming in their Computational Thinking development.

2 Background

2.1 Computational thinking frameworks

Wing in her highly cited article "Computational Thinking" defines Computational Thinking as a process that "involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (Wing, 2006). In addition to the aforementioned definition, several others appear in the literature. Some of them are closely related to programming and computing concepts, while others perceive Computational Thinking as a necessary competence both in domain specific fields and in general problem-solving skills (Tang et al., 2020).

One of the most widely adopted models closely related to programming is Brennan's and Resnick's framework (Brennan & Resnick, 2012). Based on programming with Scratch, they propose a framework that includes the following three dimensions: (a) Computational thinking concepts that correspond to programming blocks, including Sequences, Loops, Parallelism, Events, Conditionals, Operators, Data, (b) Computational thinking practices or construction processes, including Being incremental and iterative, Testing and debugging, Reusing and remixing, Abstraction and modularity and (c) Computational Thinking perspectives that reveal a shift in perspective when learning Computational Thinking, including Expressing, Connecting, Questioning.

In addition to the great effort to create frameworks that describe what Computational Thinking entails, researchers have tried to conceptualize the Computational Thinking teaching and learning process (e.g., Tikva & Tambouris, 2021a, b). Tikva and Tambouris (2021b) propose the CTPK-12 model that explains the relationships between different Computational Thinking areas such as factors, tools and learning strategies. They suggest that students' Computational Thinking development could be enhanced by proper learning strategies that are supported by appropriate tools. This study follows their recommendations for using the CTPK-12 model to design empirical studies for teaching and learning Computational Thinking and investigate some of the models' relationships.

2.2 Scaffolding strategies in Computational Thinking research

Scaffolding strategies including instructional scaffolding, adaptive, peer-, resource-scaffolding support/guidance, feedback and prompts have been explored in several studies focusing on the development of Computational Thinking (Tikva & Tambouris, 2021a). Chevalier et al. (2022) investigated the role of different types of guidance and feedback in the development of Computational Thinking. To this end, they designed an experimental study to investigate which of these methods fosters students' Computational Thinking. They explored four experimental conditions for the different combinations of with/without guidance and immediate/delayed feedback strategies. Their results support that delayed feedback could be an effective intervention method for Computational Thinking development. Angeli and Valanides (2020) investigated the impact of two scaffolding techniques, designed with gender differences into consideration. To this end, students were randomly assigned to two groups, each following a different type of scaffolding. Their findings show that both sexes benefited from both scaffolding techniques, while each gender benefited more from a different scaffolding technique. Chen et al. (2021) designed a quasi-experimental study to investigate the effects of scaffolding prompts on students' Computational Thinking. Students were assigned to three groups, each of which received cognitive prompts, metacognitive prompts and combination of cognitive and metacognitive prompts respectively. Their findings support that metacognitive scaffolding prompts could be an effective strategy to foster student's Computational Thinking. In the same line, Atmatzidou et al. (2018) explored the effects of different types of guidance (minimal vs strong) on students' metacognitive and problem-solving skills. The findings of their quasi-experimental study support that strong guidance could have a positive impact on students' metacognitive and problem-solving skills.

2.3 Attitudes towards programming/Computer Science in Computational Thinking research

Attitudes towards programming and Computer Science (CS) are of interest to Computational Thinking studies. Attitudes towards programming are explored under two major research questions: a) To what extent do specific interventions impact

students' attitudes towards programming/CS? and b) To what extent students' attitudes towards programming/CS affect their Computational Thinking? For example, Zhao and Shute (2019) measure attitudes toward CS based on a survey that includes questions about how students perceive computers such as "Computers are fun" and "Computing jobs are boring". Subsequently they explored if playing a programming video game could have an impact on students' attitudes, finding no statistically significant differences in students' attitudes before and after the intervention. They point out that the short duration of the intervention may have played a role in this outcome. In the same line, Cetin (2016) explored the effects of a Scratch-based instruction on participants' attitudes towards programming, finding no statistically significant effect. They suggest that this could be attributed to the limited duration of treatment, the participants' already high attitudes and satisfaction with the quality of teaching.

Other studies focus on how students' attitudes towards programming could affect Computational Thinking acquisition. For example, Sun et al. (2022) define programming attitude based on a framework that includes the elements of programming self-efficacy, programming utility, social needs, perceptions of programmers, and programming interest. Their results support that students' attitudes towards programming could impact their Computational Thinking, indicating them as an important factor in Computational Thinking development. Kong et al. (2018) define programming empowerment as a Computational Thinking perspective. They explore whether interest in programming and attitude towards collaboration are related to programming empowerment. Their results suggest that interest in programming could affect the acquisition of programming empowerment.

Despite the interest in attitudes towards programming/CS, there is no unanimously accepted definition by researchers. Computational Thinking studies explore various attitudes, while focusing on developing scales for them (e.g., Cetin & Ozden, 2015). Table 1 presents attitudes that appear repeatedly in the literature. In the context of this study, attitudes towards programming consist of the following three (3) dimensions: programming self-efficacy, interest in programming and programming meaningfulness.

3 Materials and methods

3.1 Purpose of the study

This study aims to investigate the effect of scaffolding programming games on the development of middle school students' Computational Thinking. To this end, we designed a scaffolding programming game named "aMazeD" based on a three-dimension scaffolding framework. The scaffolding game is aligned with Computational Thinking concepts and practices included in Brenan's and Resnik's (2012) framework. In particular, we explore how the presence of scaffolding features affects the acquisition of students' Computational Thinking. In addition, herein we

Table 1 Attitudes towards programming/CS found in the literature

Attitude		Scale item example	Study
Confidence/Self-efficacy	programming self-efficacy	I am good at programming (Kong et al., 2018)	Kukul et al., 2017; Kong et al., 2018; Durak et al., 2019
	CS self-efficacy	I feel confident about my ability to use computers (Werner et al, 2012)	Werner et al., 2012; Román-González et al., 2018
	coding confidence	I am good at coding (Mason & Rich, 2020)	Mason & Rich, 2020
	programming confidence	I am confident to learn programming (Sun et al., 2022)	Sun et al., 2022
Interest	interest in programming	I think the content of programming is fun (Kong et al., 2018)	Kong et al., 2018; Sun et al., 2022
	coding interest	Solving coding problems seems fun (Mason & Rich, 2020)	Mason & Rich, 2020
Meaningfulness/Utility	programming meaningfulness	Programming is useful to me (Kong et al., 2018)	Kong et al., 2018
	coding utility	Knowing how to code will help me to create useful things (Mason & Rich, 2020)	Mason & Rich, 2020
	programming utility	Learning programming is very useful (Sun et al, 2022)	Sun et al, 2022
Social influence/needs		My parents think coding is important (Mason & Rich, 2020)	Mason & Rich, 2020; Sun et al, 2022
Perception of coders/ programmers		I think kids who can code spend less time outdoors than other kids (Sun et al, 2022)	Mason & Rich, 2020; Sun et al, 2022

investigate the effect of students' attitudes towards programming on their Computational Thinking improvement.

3.2 Research questions

The following research questions are posed:

I Does aMazeD have a positive impact on middle school students' Computational Thinking development?
II Does aMazeD with scaffolding features have a greater impact on middle school students' Computational Thinking development than the aMazeD version without scaffolding?
III Do attitudes towards programming have an impact on middle school students' Computational Thinking?
IV Do attitudes towards programming have an impact on middle school students' Computational Thinking improvement?

3.3 Research design

In order to address the study goal, we conducted an experimental study. Ethical approval from the university ethical committee of the authors' university was obtained. In addition, all students' parents were informed and gave their consent to participate in the study. Participants were 57 students in seventh, eighth and ninth grade. From them, 29 students were randomly assigned to the experimental group where a scaffolding version of the programming game was used as the learning approach, while the rest 28 students were assigned to the control group where a version of the programming game that did not include scaffolding features was used. In order to prevent potential influence of different teachers on the outcome of the study, all students were taught by the same teacher using the same technical equipment regardless of which group they belonged to. The experiment was conducted in three phases and lasted three weeks. In the first phase, students were asked to complete a pre-test for measuring their Computational Thinking and a questionnaire measuring their attitudes towards programming. Both the pre-test and the questionnaire lasted 45 min. Students completed the pre-test and the questionnaire on two different days. In the second phase of the experiment, students participated in a 45-min intervention where they were introduced to Computational Thinking through the two versions of the programming game, depending on the group they belonged to. During the intervention, students encountered Computational Thinking concepts such as sequence, loops, conditionals and Computational Thinking practices such as testing and debugging and being incremental and iterative. Log files from the game were also collected. In the last phase, students completed a post-test for measuring their Computational Thinking which lasted 45 min.

3.4 Intervention instrument

3.4.1 The aMazeD scaffolding programming game

The "aMazeD" scaffolding programming game is based on Blockly Games: Maze and Turtle (Mousiou, 2021). In addition, the majority of the levels are based on the Computational Thinking Test (CTt) developed by Roman-Gonzalez et al. (2018). The game consists of 10 levels. Each level belongs to one of the following categories: a) Maze and b) Turtle. The player in the maze category levels uses programming blocks to guide his/her character from start to finish through a maze (Fig. 1). In the turtle category levels, the player uses programming blocks to draw the required shapes in each level.

3.4.2 Computational Thinking concepts and practices covered by the aMazeD game

The player must employ different Computational Thinking concepts and practices according to Brennan's and Resnick's framework (Brennan & Resnick, 2012) in order to solve each level. Computational Thinking concepts and Practices covered by the game are presented in Table 2.

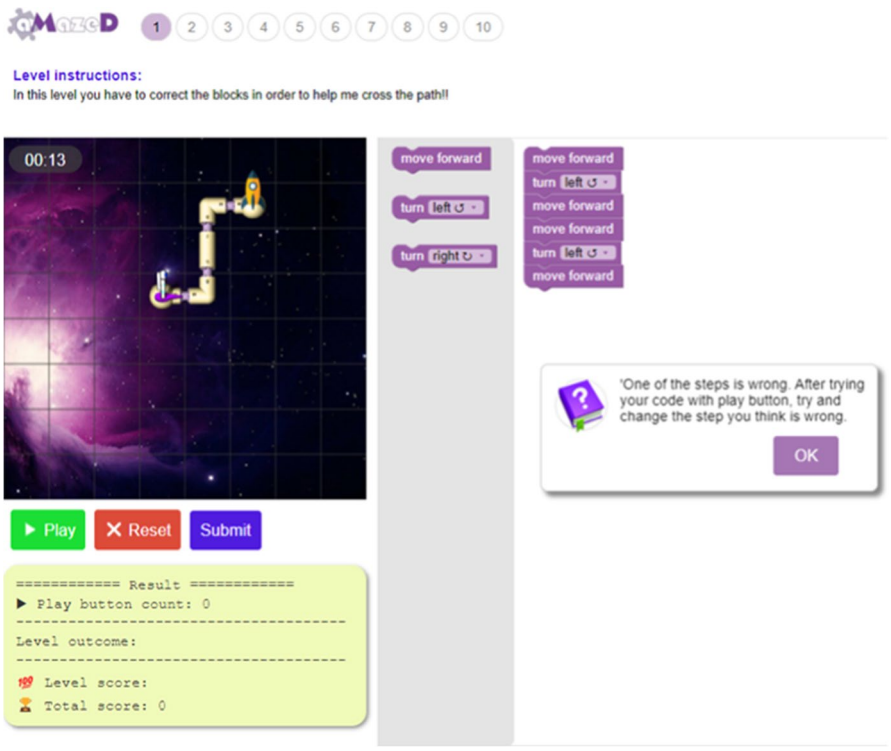

Fig. 1 The aMazeD scaffolding programming game

Table 2 CT Concepts and practices per aMazeD level

Concept	Description	Application to aMazeD levels
Computational Thinking concepts adopted from Brennan and Resnick (2012)		
Sequences	Basic instructions and directions	The player needs to design a sequence of steps in order to solve the level (Level 1,7)
Loops	Repeat a set of instructions for a specific number of times or until a condition becomes true	The player needs to repeat a set of instructions in order to solve the level (Level 2–6, 8–10)
Conditionals	Constraints that allow the execution of different instructions	The player needs to design a solution that involves the selection of a choice based on constraints (4–6)
Computational Thinking practices adopted from Brennan and Resnick (2012)		
Testing and debugging	Trial and error processes for correcting malfunctions	The player needs to make corrections to a given set of instructions (Level 1–3, 5, 7)
Being incremental and iterative	Design and implement solutions using iterative processes	The player uses the play button in order to see the execution of the game and make changes to his/her solution until the final submission (Level 1–10)

3.4.3 aMazeD scaffolding version

The scaffolding version of the aMazeD game is designed and developed to support scaffolding, based on a three-dimension framework that includes: i) the provision of a semi-finished or semi-correct solution, ii) instructions and explanations of the Computational Thinking concepts required for the solution of each level and iii) the provision of support regarding the logic behind the solution design. The provision of semi-finished programs aims to facilitate the understanding and use of Computational Thinking concepts by students through making small changes to the program instructions (Werner et al., 2012). The player can execute the semi-finished programs and observe exactly how the character behaves. In this way, he/she has a better and more complete understanding of how sequence, loops and conditionals work. After the player clicks the play button for the first time, an explanation of the Computational Thinking concepts covered in the level is displayed. After the player clicks the play button a second time, a prompt about the logic behind the solution of the level is provided to help students build comprehension of how they could employ Computational Thinking concepts to solve the level. In this way, we build scaffolding for students by first ensuring the comprehension of Computational Thinking concepts by providing them with semi-finished solutions and explanations regarding the use of the concepts. Subsequently, we provide support to students to help them understand how they could use these concepts in order to design effective solutions.

Each level is designed based on the three-dimension framework described above. In the following paragraph we present how the aforementioned framework is applied at level two. The level starts with the semi-finished program pre-loaded at the level workspace (Fig. 2) and the instruction for correcting the given blocks in order to enable the character to reach the rocket. After the player clicks the play button for the first time, the following explanation about the "repeat until" block is displayed: "Repeat" is used to execute one or more blocks more than once. The next time the player runs the program by clicking the play button, the following prompted is

Fig. 2 Level 2 semi-finished instructions

displayed: “How many steps must the character take before turning left? How many does he/she take now? ".

In contrast, the non-scaffolding version does not provide students with semi-finished solutions, prompts or explanations.

3.5 Data collection

In this study, we measured students' pre-intervention and post-intervention Computational Thinking using the CTtest. The CTtest was developed and validated by Román-González et al. (2018). It is a direct assessment method that is widely accepted as a reliable way to measure Computational Thinking. The CTtest consists of 28 multiple choice items. Questions are presented using the interface of Maze or Canvas and the answers are presented as visual arrows or blocks.

We also collected the aMazeD log files that include the following information for each student: a) the success or failure in each level and b) the code submitted for each level.

An instrument for measuring attitudes towards programming was adapted from Kong et al. (2018). We used the following three constructs of the aforementioned instrument translated in the students' native language: programming meaningfulness, programming self-efficacy and interest in programming to measure students' attitudes towards programming. The scale consists of 13 items and students were asked to indicate their level of agreement with each item on a 5-point Likert scale (1 = Strongly agree; 5 = Strongly disagree).

4 Results

4.1 Demographics

57 students whose parents gave their consent to participate in the study were randomly assigned to the control and experimental group. There were 5 students from the control group and 7 from the experimental group who were absent either during the completion of the tests or during the intervention. This resulted in a final sample of 45 students, of whom 23 belong to the control group and the rest 22 to the experimental group. The distribution of students by grade and gender is shown in Table 3. Among participants, 23 (51%) students were male and 22 (49%) were female. 13 (29%) were in 7th grade, 21 (47%) were in 8th grade and 11 (24%) were in 9th grade.

Table 3 Distribution of participants by grade and gender

		Grade			Gender	
		7th	8th	9th	Male	Female
Version	Non-Scaffolding	7	10	6	14	9
	Percentage in the non-scaffolding group	30.4%	43.5%	26.1%	60.9%	39.1%
	Scaffolding	6	11	5	9	13
	Percentage in the scaffolding group	27.3%	50%	22.7%	40.9%	59.1%

4.2 CTtest

CTtest (Román-González et al., 2018) was employed to measure CT pre-intervention and post-intervention scores. For each item we assigned 1 if it was correct and 0 if it was incorrect. The score for each test ranged from 0 to 28. The scale had an acceptable level of internal consistency, as determined by a Cronbach's alpha of *0.763* reported in the pre-intervention data and an acceptable level of internal consistency as determined by a Cronbach's alpha of *0.803* reported in the post-intervention data.

4.3 Analytics

We calculated the overall game score for each student based on aMazeD game logs. For each level we assigned 1 if it was successfully completed and 0 otherwise. The overall game score for each student ranged from 0 to 10. The Cronbach's alpha coefficient was 0.753. We also calculated the following scores based on the inspection of the submitted code:

a) Conditional-Level and Loop-Level score. We assigned 1 for each successfully completed level belonging to the "Conditionals" concept (Table 1) and 0 otherwise. The overall Conditional-Level score for each student ranged from 0 to 3. Accordingly, we assigned 1 for each successfully completed level belonging to the "Loops" concept (Table 1) and 0 otherwise. The overall Loop-Level score for each student ranged from 0 to 8.
b) Conditional-Use and a Loop-Use score. We assigned 1 if the submitted code contained Conditionals for each correctly completed level belonging to the "Conditionals" concept and 0 otherwise. The overall Conditional-Use score for each student ranged from 0 to 3. Accordingly, we assigned 1 if the submitted code contained Loops for each correctly completed level belonging to the "Loops" concept and 0 otherwise. The overall Loop-Use score for each student ranged from 0 to 8.
c) Conditional-Ratio and Loop-Ratio. We calculated the Conditional-Ratio as the ratio between Conditional-Use score and Conditional-Level Score and the Loop-Ratio as the ratio between Loop-Use score and Loop-Level score.

4.4 Scale of attitudes towards programming

A scale adapted from Kong et al. (2018), was used to measure student's attitudes towards programming. The scale consisted of 13 items 5-point Likert scale, (1 = Strongly agree and 5 = Strongly disagree). The score of each student was calculated as the sum of the 13 items and ranged from 13 to 65. 40 of the participants were filled in the attitudes towards programming scale. The scale had a high level of internal consistency, as determined by a Cronbach's alpha of *0.948* (Table 4). We classified the participants into three groups based on their percentile value in the scale score distribution: Low-attitudes towards programming students ($n = 13$), Moderate-attitudes towards programming ($n = 14$) and High-attitudes towards programming students ($n = 13$).

Table 4 Internal consistency of the scale of Attitudes towards Programming

Construct	Number of items	Cronbach's alpha
programming meaningfulness	4	0.921
programming self-efficacy	5	0.912
interest in programming	4	0.900
entire scale	13	0.948

4.5 Does aMazeD have a positive impact on middle school students' CT development?

The first research question was, "Does aMazeD have a positive impact on middle school students' CT development?" Our hypothesis was that aMazeD would have a positive impact on middle school students' CT development. A paired-samples t-test was used to determine whether there was a statistically significant mean difference between the pre-intervention CT scores and the post-intervention CT scores of the students. No outliers were detected. The assumption of normality was not violated, as assessed by Shapiro–Wilk's test *(p = 0.612)*. We found a significant mean increase of *3.933, 95% CI [3.097, 4.769], t(44) = 9.481,p < 0.001* between pre-intervention and post-intervention CT scores, with a large effect size (*Cohen's d = 1.413*). Students CT post-intervention scores were higher (*M = 19.333, SD = 4.772*) compared to their CT pre-intervention scores (*M = 15.4, SD = 4.653*). This result supports our hypothesis that aMazeD would have a positive impact on students' CT development.

4.6 Does aMazeD with scaffolding features have a greater impact on middle school students' CT development than the aMazeD version without scaffolding features?

The second research question was "Does aMazeD with scaffolding features have a greater impact on middle school students' CT development than the aMazeD without scaffolding?". Our hypothesis was that the scaffolding version of aMazed would have a greater impact on students' CT development. CT pre-scores and post-scores were measured by the CTtest (Román-González et al., 2018). An independent t-test showed that the mean of the pre-test CT scores of the scaffolding group was not significantly higher (*M = 15.727, SD = 4.442*) than that of the non-scaffolding group (*M = 15.087, SD = 4.926*); *t (43) = − 0.457, p = 0.650*. Thus, we can conclude that the two groups were equivalent in terms of students' CT scores prior to the intervention. An ANCOVA was run to determine the effect of the scaffolding version of the game on post-intervention CT scores after controlling for pre-intervention CT scores. There was a linear relationship between pre-intervention CT scores and post-intervention CT scores for each group, as assessed by visual inspection of a scatter plot. There was homogeneity of regression slopes as the interaction term was not statistically significant, *F(1,41) = 0.180, p = 0.673*. Standardized

residuals for the interventions and for the overall model were normally distributed, as assessed by Shapiro–Wilk's test ($p > 0.05$). There was homoscedasticity and homogeneity of variances, as assessed by visual inspection of a scatterplot and Levene's test of homogeneity of variance ($p = 0.911$), respectively. There were no outliers in the data, as assessed by no cases with standardized residuals greater than ± 3 standard deviations. After adjustment for pre-intervention CT scores, there was a statistically significant difference in post-intervention CT scores between the scaffolding and the non-scaffolding group, $F(1,42) = 5.657$, $p = 0.022$.

We further analyze students' log files. Mann–Whitney U test was run to determine if there were differences in Conditional-Use scores between the non-scaffolding and scaffolding group. Distributions of the Conditional-Use scores for the two groups were not similar, as assessed by visual inspection. Conditional-Use scores for the scaffolding group (*mean rank* = *29.30*) were statistically significantly higher than for the non-scaffolding group (*mean rank* = *16.98*), $U = 391.5$, $z = 3.409$, $p = 0.001$. Respectively, Mann–Whitney U test was run to determine if there were differences in Loop-Use Score between the non-scaffolding and scaffolding group. Distributions of the Loop-Use Scores for the two groups were not similar, as assessed by visual inspection. Loop-Use scores for the scaffolding group *(mean rank* = *30.27*) were statistically significantly higher than for the non-scaffolding group (*mean rank* = *16.04*), $U = 413$, $z = 3.695$, $p < 0.001$.

4.7 Do attitudes towards programming have an impact on students' CT?

The third research question was "Do attitudes towards programming have an impact on middle school students' CT? ". Our hypothesis was that positive attitudes towards programming would have a greater impact on students' CT scores. A one-way ANOVA was conducted to determine if the students' CT pre-test scores were different for the low/moderate/high attitudes groups. There were no outliers, as assessed by boxplot; data was normally distributed for each group, as assessed by Shapiro–Wilk test ($p > 0.05$); and there was homogeneity of variances, as assessed by Levene's test of homogeneity of variances ($p = 0.818$). CT pre-test score increased from low ($M = 13.769$, $SD = 4.902$) to moderate ($M = 15.429$, $SD = 4.327$) to high ($M = 17{,}154$, $SD = 4.793$) attitudes group, in that order, but the differences between attitudes groups was not statistically significant, $F(2,37) = 1.706$, $p = 0.196$. This result does not support the hypothesis that student's attitudes towards programming would have an impact on middle school students' CT.

4.8 Do attitudes towards programming have an impact on students' CT improvement?

The fourth research question was "Do attitudes towards programming have an impact on students' CT improvement?". Our hypothesis was that attitudes towards programming would have an impact on students' CT development. A one-way

ANOVA was conducted to determine if the changes in students' CT scores were different for the low/moderate/high attitudes groups. There were no outliers, as assessed by boxplot; data was normally distributed for each group, as assessed by Shapiro–Wilk test ($p > 0.05$); and there was homogeneity of variances, as assessed by Levene's test of homogeneity of variances ($p = 0.113$). Changes in CT scores increased from moderate ($M = 3.143$, $SD = 3.348$), to high ($M = 3.539$, $SD = 1.808$), to low ($M = 4.462$, $SD = 2.817$) attitudes group, but the differences were not statistically significant, $F(2,37) = 0.807$, $p = 0.454$. This result does not support the hypothesis that student's attitudes towards programming would have an impact on middle school students' CT development.

5 Discussion

Our first hypothesis was that aMazeD would have a positive impact on middle school students' CT. Data analysis and results seem to support this hypothesis. Participants significantly improved their CT scores at the CTtest after playing the aMazeD. This is consistent with prior research showed that playing programming games could improve students' Computational Thinking (e.g., Hooshyar et al., 2021; Zhao & Shute, 2019). However, since this is a one-group pretest–posttest design, it cannot be excluded that the differences between the pre-test and post-test are due to threats such as maturation (Fraenkel et al., 2012).

The second hypothesis was that aMazeD with scaffolding features would have a greater impact on middle school students' CT than the aMazeD version without scaffolding features. Both groups experienced an improvement in their post-intervention CT scores, but students who played the scaffolding version of the game had significantly higher CT post-scores (Table 5). Furthermore, students in the scaffolding group not only did better on the post-test, but they had significantly higher Conditional-Use and Loop-Use scores (Table 6). The code they submitted to the game was of higher quality and included the use of Conditionals and Loops. It is indicative that students in the scaffolding group who used conditionals in all successful levels belonging to the "Conditional Concept" concept amount to 18 out of 22 compared to 6 out of 23 students in the non-scaffolding group. Respectively, students in the scaffolding group who used loops in all successful levels belonging to the "Loop Concept" amount to 18 out of 22 compared to 4 out of 23 students in the non-scaffolding group. These results suggest that scaffolding could be an effective learning technique for developing students' CT and help them understand the core concepts

Table 5 Computational Thinking pre-scores and post-scores means by game version

Game Version	Means of Pre-intervention Scores	Means of Post-intervention Scores	Means of CT scores changes
Scaffolding version	15.727	20.546	4.818
Non-Scaffolding version	15.087	18.174	3.087

Table 6 Computational Thinking Conditional-Level, Loop-Level, Conditional-Use, Loop-Use scores, Conditional-Ratio and Loop-Ratio means by game version

Game Version	Means of Conditional-Level Scores [0–3]	Means of Loop-Level Scores [0–8]	Means of Conditional-Use Scores [0–3]	Means of Loop-Use Scores [0–8]	Means of Conditional-Ratio	Means of Loop-Ratio
Scaffolding version	2.86	6.05	2.50	5.36	0.871	0.878
Non- Scaffolding version	2.57	4.65	1.13	2.22	0.384	0.409

of CT such as Conditionals and Loops. Prior research also shows results regarding the relationship between scaffolding and CT development. Studies conclude that scaffolding could have a positive impact on CT development. Specifically, Chen et al. (2021) findings of their quasi-experimental study revealed that metacognitive prompts significantly improved students' CT outcomes. In the same line, Angeli and Valanides (2020) found that students who participated in their study benefited from the scaffolding techniques used. Furthermore, Chevalier et al. (2022) found that students in their study benefited from guidance and feedback learning methods.

The third hypothesis was that attitudes towards programming would have an impact on students' CT scores. No significant differences were found between the three groups (low/moderate/high) in the results of students' CT pre-tests. Although students' pre-test scores were very similar in general, as shown in Fig. 3, the students of the low attitudes group were less successful than students in the moderate and high attitudes group. Previous studies indicate that Computational Thinking is related with attitudes towards programming (Sun et al., 2022) and suggest that interest in programming could be an important factor in the acquisition of CT (Kong et al., 2018), proposing interest-driven strategies for CT teaching and learning (Kong, 2016).

The fourth hypothesis was that attitudes towards programming would have an impact on students' CT development. Although this hypothesis was not confirmed as

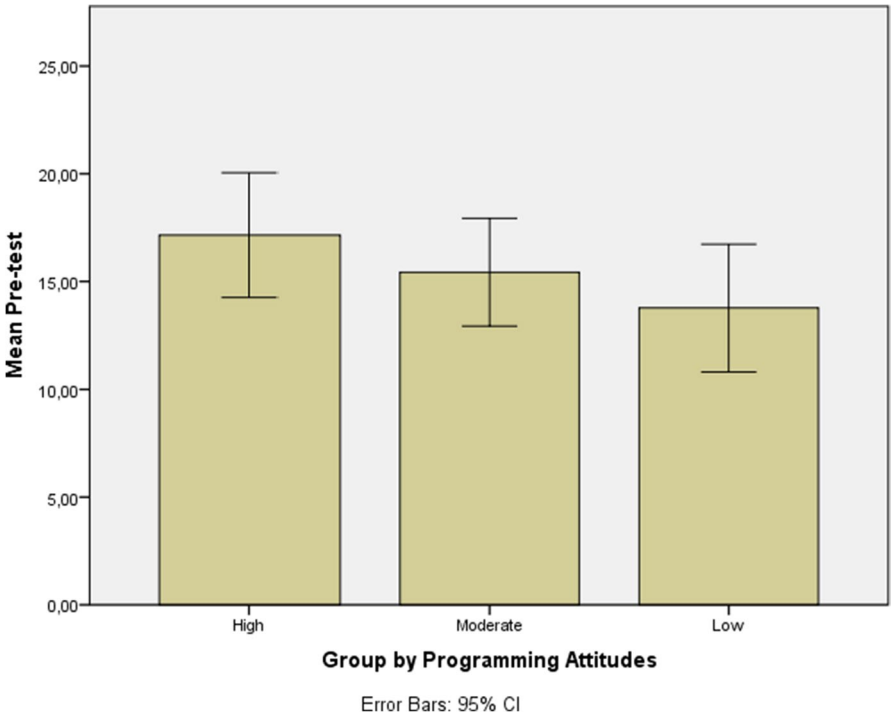

Fig. 3 Means of pre-tests scores by attitudes towards programming group

Table 7 Computational Thinking changes in pre-scores and post-scores means by game version and attitudes towards programming group

Game Version	Attitudes towards programming Group	Means of Change in CT Scores
Non-scaffolding version	High	4.000
	Moderate	2.556
	Low	1.600
Scaffolding version	High	3.000
	Moderate	4.200
	Low	6.250

no significant differences were found between the three groups (low/moderate/high) in students' CT improvement, the descriptive statistical analysis reveals interesting results. As shown in Table 7, changes in students' CT scores for the non-scaffolding version increase from low ($M=1.600$, $SD=0.872$) to moderate ($M=2.556$, $SD=1.069$), to high attitudes group ($M=4.000$, $SD=5.35$) (Fig. 4). This result is consistent with other studies (Sun et al., 2022) which have shown that students with negative attitudes

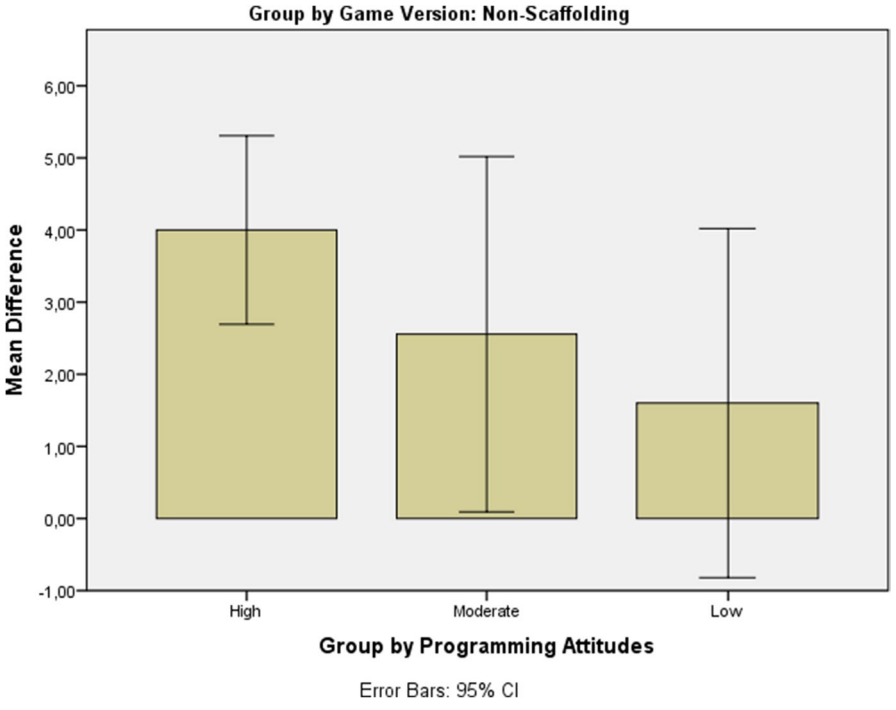

Fig. 4 Means of score changes by attitudes towards programming group for the non-scaffolding group

towards programming may find it more difficult to develop their Computational Thinking than students with positive attitudes towards programming. Results indicate that students are struggling to develop their Computational Thinking skills when they are not provided with an appropriate learning strategy. This is in line with previous studies which suggest that students face great difficulties without proper guidance (Denner et al., 2012). However, this is not the case for students that experienced the scaffolding version. Changes in students' CT scores in the scaffolding version increase from to high ($M=3.000$, $SD=0.894$) to moderate ($M=4.200$, $SD=1.655$) to low ($M=6.250$, $SD=0.491$) attitudes group (Fig. 5). This result could have important implications in the design of appropriate learning interventions regarding the choice of the learning strategies in relation to students' attitudes towards programming. Results suggest that students with low and moderate attitudes towards programming tend to benefit more from the scaffolding strategy than students with higher attitudes towards programming. The provision of scaffolding through semi-finished programs and prompts could engage students who tend to have low interest in programming and low programming self-efficacy, by reducing difficulty levels and providing effective supplies for developing Computational Thinking.

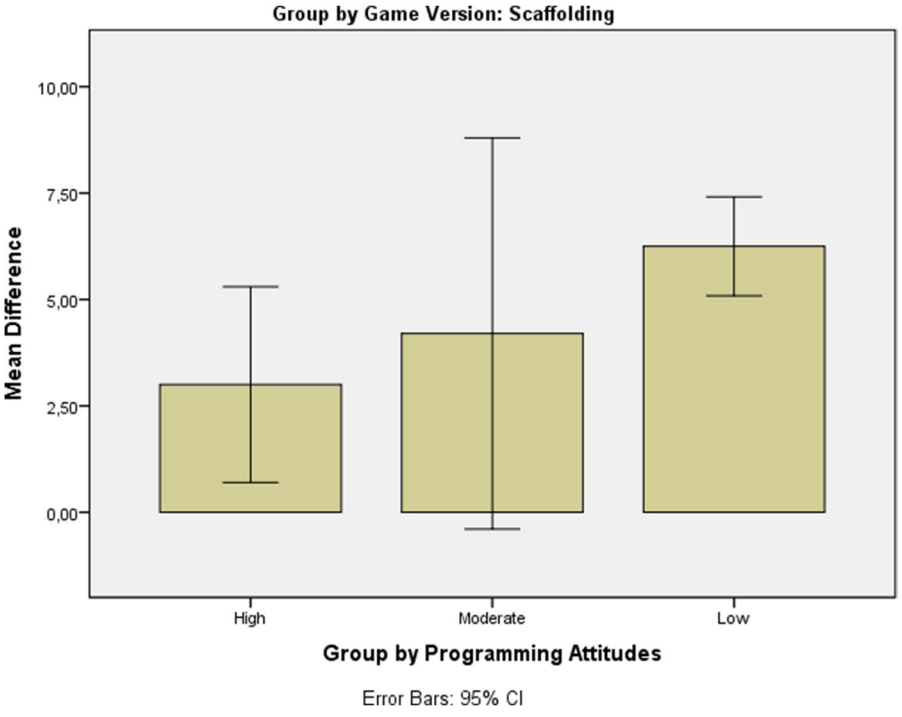

Fig. 5 Means of score changes by attitudes towards programming group for the scaffolding group

6 Conclusions

This study explores the effect of scaffolding programming games on the development of middle school students' Computational Thinking. In addition, herein we explore the effect of students' attitudes towards programming on their Computational Thinking. Students were introduced to Computational Thinking under two distinct experimental conditions: a scaffolding version of a programming game and a non-scaffolding version of the same game. Results report statistically significant learning gains between the pre-intervention and post-intervention CT scores for all students and statistically significant improvement in learning outcomes in favour of the scaffolding group. Furthermore, students in the scaffolding group not only showed better learning outcomes overall, but also submitted higher quality code in terms of using conditionals and loops during the game. The findings support that scaffolding helps students develop Computational Thinking and deepen their understanding of the related concepts. In addition, the study hypothesized that attitudes towards programming would have an impact on students' Computational Thinking and Computational Thinking development. However, this hypothesis was not confirmed from the results that report a non-statistically significant difference in both cases. Nevertheless, students' Computational Thinking in the non-scaffolding group found to be higher for students with a more positive attitude towards programming. Specifically, students in the high attitudes group had greater learning gains, followed by students in the moderate attitudes group and students in the low attitudes group for the non-scaffolding version of the game. On the other hand, students in the low attitudes group had greater learning gains, followed by students in the moderate attitudes and students in the high attitudes group for the scaffolding version of the game.

The implication of these findings is important, as they provide support that scaffolding in computational thinking games could be an effective strategy for teaching and learning computational thinking to middle school students fostering a deeper understanding of Computational Thinking concepts. In addition, when it comes to students' attitudes towards programming, students who perceive programming as less meaningful, less interesting and have lower programming self-efficacy could particularly benefit from scaffolding aspects in programming games.

However, this study has some limitations including the small sample size and the short duration of the intervention. A longer duration could provide more insights on students' learning gains. In addition, we based our analysis only on tests, questionnaires and logs. Including interviews and video recording could have provided a more holistic understanding of students' CT development. The inclusion of students from a single school could be also considered as a limitation of the study.

Acknowledgements We would like to thank Marcos Román-González who shared with us the full version and the specification table sheet of the Computational Thinking Test (CTt) and Maria Mousiou for her contribution to the development of the scaffolding game.

Funding Open access funding provided by HEAL-Link Greece.

Data availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Code availability Not applicable.

Declarations

Conflicts of interest/Competing interests The authors report there are no competing interests to declare.

References

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, *105*.https://doi.org/10.1016/j.chb.2019.03.018

Atmatzidou, S., Demetriadis, S., & Nika, P. (2018). How Does the Degree of Guidance Support Students' Metacognitive and Problem Solving Skills in Educational Robotics? *Journal of Science Education and Technology, 27*(1), 70–85. https://doi.org/10.1007/s10956-017-9709-x

Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment. *User Modeling and User-Adapted Interaction, 27*, 5–53. https://doi.org/10.1007/s11257-017-9187-0

Brennan, K., & Resnick, M. (2012). New frameworks for thinking. *Annual American Educational Research Association Meeting, Vancouver, BC, Canada*, 1–25. http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf. Accessed 15 Oct 2020.

Cetin, I., & Ozden, M. (2015). Development of computer programming attitude scale for university students. *Computer Applications in Engineering Education, 23*, 667–672. https://doi.org/10.1002/cae.21639

Cetin, I. (2016). Preservice Teachers' Introduction to Computing: Exploring Utilization of Scratch. *Journal of Educational Computing Research, 54*(7), 997–1021. https://doi.org/10.1177/0735633116642774

Chen, C. H., Liu, T. K. and Huang, K. (2021). 'Scaffolding vocational high school students' computational thinking with cognitive and metacognitive prompts in learning about programmable logic controllers', *Journal of Research on Technology in Education*, 0(0), pp. 1–18.https://doi.org/10.1080/15391523.2021.1983894

Chevalier, M., et al. (2022). The role of feedback and guidance as intervention methods to foster computational thinking in educational robotics learning activities for primary school. *Computers and Education, 180*, 104431. https://www.sciencedirect.com/science/article/pii/S0360131522000021

Clark, D. B., & Sengupta, P. (2019). Reconceptualizing games for integrating computational thinking and science as practice: collaborative agent-based disciplinarily-integrated games. In *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1636071

de Souza, A. A., Barcelos, T. S., Munoz, R., Villarroel, R., & Silva, L. A. (2019). Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops. *IEEE Access, 7*, 82848–82866. https://doi.org/10.1109/access.2019.2924343

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education, 58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

Durak, H. Y., Yilmaz, F. G. K., & Bartin, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology*, *10*(2), 173–197. https://doi.org/10.30935/cet.554493

Fraenkel, J. R., Wallen, N. E., & Hyun, H. H. (2012). *How to design and evaluate research in education* (8th ed.). Mc Graw Hill.

García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior, 80*, 407–411. https://doi.org/10.1016/j.chb.2017.12.005

Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: Exploring computational thinking through code analysis.*Interactive Learning Environments*, *26*, 386–401. https://doi.org/10.1080/10494820.2017.1337036

Garneli, V., & Chorianopoulos, K. (2019). The effects of video game making within science content on student computational thinking skills and performance. *InteractiveTechnology and Smart Education*. https://doi.org/10.1108/ITSE-11-2018-0097

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*, *126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Heintz, F., Mannila, L., & Farnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education.*Proceedings - frontiers in education conference, FIE*. https://doi.org/10.1109/FIE.2016.7757410

Israel-Fishelson, R., & Hershkovitz, A. (2020). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. *Journal of Educational Computing Research, 58*(5), 891–918. https://doi.org/10.1177/0735633119887187

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computationalthinking education. *Computers and Education*, *127*, 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

Kong, S.-C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers and Education*, *151*. https://doi.org/10.1016/j.compedu.2020.103872

Kukul, V., Gₔokçearslan, S¸., & Günbatar, M. S. (2017). Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation and reliability. *Educational Technology Theory and Practice, 7*(1), 158–179. https://doi.org/10.17943/ETKU.72918

Mason, S. L., & Rich, P. J. (2020). Development and analysis of the elementary student coding attitudes survey. *Computers & Education, 153*, 103898. https://doi.org/10.1016/j.compedu.2020.103898

Mousiou, M. (2021). Developing a Computational Thinking Environment through Learning Programming [Master's thesis, Hellenic Open University]. Hellenic Open University Research Repository.https://apothesis.eap.gr/handle/repo/54054

Repenning, A., Grover, R., Gutierrez, K., Repenning, N., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Horses, I. H. M., Basawapatna, A., & Gluck, F. (2015). Scalable Game Design. *ACM Transactions on Computing Education, 15*(2), 1–31. https://doi.org/10.1145/2700517

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior, 80*, 441–459. https://doi.org/10.1016/j.chb.2017.09.030

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-basedcomputation: A theoretical framework. *Education and Information Technologies*, *18*, 351–380. https://doi.org/10.1007/s10639-012-9240-x

Sun, L., Hu, L., & Zhou, D. (2022). Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience. *Computers and Education, 181*(27), 104457. https://doi.org/10.1016/j.compedu.2022.104457

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies.*Computers & Education*, *148*. https://doi.org/10.1016/j.compedu.2019.103798

Tikva, C., & Tambouris, E. (2021a). A systematic mapping study on teaching and learning Computational Thinking through programming in higher education. *Thinking Skills and Creativity*, *41*(December 2020), 100849. https://doi.org/10.1016/j.tsc.2021.100849

Tikva, C., & Tambouris, E. (2021b). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education, 162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *SIGCSE'12 -Proceedings of the 43rd ACM technical symposium on computer science education*. https://doi.org/10.1145/2157136.2157200

Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning, 6*, 1–17. https://doi.org/10.4018/IJGBL.2016010101

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society a: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers and Education, 141*(July), 103633. https://doi.org/10.1016/j.compedu.2019.103633