



# The effect of PDCA cycle strategy on pupils' tangible programming skills and reflective thinking

Xin Gong<sup>1</sup> · Shufan Yu<sup>2</sup> · Jie Xu<sup>3</sup> · Ailing Qiao<sup>1</sup> · Han Han<sup>4</sup>

Received: 10 April 2023 / Accepted: 5 July 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

## Abstract

Tangible programming combines the advantages of object manipulation with programmable hardware, which plays an essential role in improving programming skills. As a tool for ensuring the quality of projects and improving learning outcomes, the PDCA cycle strategy is conducive to cultivating reflective thinking. However, there is still a lack of empirical research on the effect of introducing the PDCA cycle strategy into programming education. In this study, using a PDCA cycle strategy, in a four-pronged model of “(P)draw up a plan, (D)assemble and programming, (C)test and debug, display and reflect (A),” and its effects on students' programming skills and their reflective thinking were explored. There were 65 children between the ages of 7 and 8 years participated in this study. There were 31 students in each of the experimental group and the control group. A combination of qualitative and quantitative research methods was adopted in this research, and students' programming processes and results were observed and counted. The study results revealed that after attending the ‘Magic Card Robot’ course that applied the PDCA cycle strategy, the experimental group students outperformed their counterparts in programming skills (sequencing, repetitive and conditional structures). Meanwhile, the experimental group students' reflective thinking levels were higher than those of the control group students. These findings imply that tangible programming education using the PDCA cycle strategy in the course has potential.

**Keywords** Tangible programming · Programming skills · Reflective thinking · PDCA cycle strategy

## 1 Introduction

Programming is essential in improving computational thinking (Chen et al., 2023) and problem-solving capabilities and is considered one of the ideal ways to develop 21st-century skills (Hsu et al., 2018). In this digital era, programming

---

✉ Shufan Yu  
yushufan1993@gmail.com

Extended author information available on the last page of the article

has permeated various aspects of our daily life (Iivari et al., 2020). In the educational field, programming instruction has flourished, particularly among young pupils. However, due to the obscure syntax and logic of computer programming languages (Demir, 2022), many novices struggle to code independently and solve practical problems. This can lead to a lack of self-confidence and a decreased interest in learning programming (Yang et al., 2023). Additionally, in most programming classes, students interact primarily with computer screens (Alonso, 2020; Su et al., 2022), which is not an ideal way for young students with developing eyesight to learn programming (Marsh et al., 2016). As a result of the ongoing conflict between children's programming education and educational teaching demands, tangible programming aids driven by tangible programming have evolved as solutions to those issues (Wyeth, 2008).

Tangible programming combines the advantages of object manipulation with programmable hardware that assists students in reducing cognitive load (Sapounidis et al., 2015) and mastering extremely abstract concepts (Zhong et al., 2022). In addition, tangible programming requires continuous iteration for dynamic quality improvement, which involves not only debugging the code but also designing and assembling the robot. Reflective thinking plays a vital role in debugging and problem-solving (Yildiz Durak, 2020), which also has been shown to help children retain knowledge longer (Tan, 2021).

To ensure effective programming instruction, researchers have put various strategies into practice, such as paired programming (Sun et al., 2022) and collaborative learning (Lai & Wong, 2022). However, these strategies mostly emphasize the organizational form of programming learning and neglect the value of reflection in programming. Although some researchers have introduced reflection into teaching, it is mainly utilized as a post-evaluation method (Lin et al., 2022; Shanley et al., 2022) instead of an essential part of the teaching strategy. The Plan-Do-Check-Action (PDCA) cycle proposed by Walter A. Shewhart (Deming, 2000) views 'check' as a crucial step that assures a spiral of high-quality learning by highlighting the value of reflection, which offers a possibility to develop children's reflective thinking in tangible programming education (Hellberg & Fauskanger, 2022). In this regard, this paper aims to explore whether the integration of the PDCA cycle strategy can facilitate the pupils' tangible programming performance, a quasi-experiment was conducted among Chinese elementary school students, and several instruments were utilized to assess their programming skills and reflective thinking levels.

## 2 Literature review

### 2.1 Tangible programming learning

Tangible programming is a form of programming that uses programmable hardware to generate human-computer interaction, and the process of running a physical object can be described as program execution (Schweikardt & Gross, 2008). Moreover, tangible programming provides real-time feedback based on the operation of the robot, distributes programming errors across physical hardware, clarifies the relationship

between code faults and tangible coding mechanisms (Silvis et al., 2022), and visualizes and concretizes abstract concepts (Bers & Horn, 2010). According to Piaget's theory of cognitive development (Piaget, 1973), physical teaching aids are necessary to help children understand abstract concepts during the concrete operations stage (Burleson et al., 2018; Revelle et al., 2005) and to train their logical thinking (Ackermann, 1996; Piaget, 1959). In this regard, tangible programming lowers children's threshold to learn complex computer syntax. Sapounidis et al. (2015, 2019) have conducted several studies about tangible learning, and the results showed that it was attractive to young girls and supported a high level of exploration. Additionally, in Cejka et al. (2006)'s study, children as young as four years old could construct simple robotic projects and understand abstract concepts by manipulating programmable hardware. The effectiveness of tangible programming in teaching has been confirmed by many researchers.

As the use of tangible programming in children's programming language learning grows, the development of tangible programming aids has become a topic of interest in elementary programming education (Fischer & Lau, 2006; Marshall, 2007). For example, Perlman (1976) developed a Slot Machine that controls robot motion by inserting cards. In Bers et al. (2019)'s study, Kibo is another tangible interface programming tool that combines Lego blocks in the correct order. However, while the diversity and practicality of tangible programming aids have significantly increased, their usage in teaching practice has primarily focused on debugging the order of programmable hardware and less on improving the construction-based robot regarding its design and assemble flaws (Silvis et al., 2022; Smith, 2009). In addition, existing studies on children's robotics teaching environments did not make a clear distinction between issues of programming (e.g., omitting a command) and issues of robot assembly (e.g., connecting a sensor to the wrong port) (Socratous, 2020). For example, Bers et al. (2014) used construction-based robotics to train children's computational thinking and emphasized the need to set aside more time to think about computer programs rather than spend time assembling robots. Moreover, in Feijoo-Almonacid and Rodriguez-Garavito's (2022) study, a robot Eli was provided that was easy to assemble. Thus, children would focus more on debugging the robot's movement programming.

According to the abovementioned studies, fixing robot physical problems has not received as much attention as debugging programming breakdowns. If their first forays into programming, students rarely succeed in producing acceptable solutions (Chen et al., 2020). Therefore, it is crucial to develop children's programming skills by simultaneously grappling with bugs in the program and the physical aids (Yildiz Durak, 2020).

## 2.2 Debugging and reflective thinking

According to Dewey's empiricism (Yürük, 2007), reflective thinking is crucial for effective problem-solving (Antonio, 2020; Bayrak & Usuel, 2011; Kizilkaya & Aşkar, 2009), and the key to debugging is to continuously identify and correct problems. For this reason, it is obvious that reflective thinking can help children's

continuous debugging of code and robots in programming learning (Yildiz Durak, 2020), thereby enhancing programming skills. Specifically, in terms of debugging programming code, reflective thinking can help identify complex problems (Altın & Saracaloğlu, 2018; Liao & Wang, 2019), enabling children to manage their programming process, question their decisions and actions, and explore alternative solutions to improve their programs' quality (Havenga et al., 2013). In terms of modifying physical robots, assembling LEGO robots can be viewed as executing a design, and reflective thinking is an integral part of this process (Hong & Choi, 2019; Walther et al., 2011). Designers use reflective thinking to review previous experiences and select appropriate solutions to reorganize and rearrange the design work (Salido & Dasari, 2019). Therefore, reflective thinking is considered the key to achieving successful debugging code and modifying robots in learning tangible programming.

However, most empirical studies on tangible programming have focused on the development of computational thinking skills such as algorithmic thinking (Evripidou et al., 2021) and problem-solving (Shim et al., 2017), and there is a dearth of research focusing on reflective thinking (Angeli & Valanides, 2020; Cho & Lee, 2017; Hsieh et al., 2022). In terms of programming instructional approaches, existing studies cared more about passive post-reflective correction by children (Lin et al., 2022; Malik et al., 2021; Shanley et al., 2022). Although some studies have used reflective learning during the “check and compare final procedures” in the teaching process (Burlison et al., 2018), they do not explicitly include reflection as a specific part of the teaching strategy. Consequently, there is a need to explore teaching strategies that can develop children's iterative reflection ability, thereby providing the right ideas for the effective application of tangible programming aids and teaching practices.

### 2.3 Application of PDCA cycle strategy in programming

The PDCA cycle was proposed by Shewhart (1931), and it has been commonly used as a problem-solving model in the field of quality management (Choo et al., 2007). PDCA emphasizes continuous improvement learning and recognizes reflection as a critical step (Hellberg & Fauskanger, 2022). The teaching strategies of PDCA can be categorized into four stages. Firstly, in the “P (plan)” stage, solutions are developed based on the requirements and objectives of the problem. Secondly, in the “D (do)” stage, the solutions are implemented. Thirdly, in the “C (check)” stage, the implementation process is closely monitored to identify any problems. Lastly, in the “A (act)” stage, the reasons for failure are analyzed, and the successes are used as a standard to continuously enhance the quality of the product.

During instruction, the PDCA cycle strategy is commonly used for the management and monitoring of teaching quality. In the study of Walasek et al. (2011), the PDCA cycle strategy was a valuable tool for ensuring the quality of e-learning projects and improving student learning outcomes. Blagojević and Micić (2013) applied the PDCA cycle in an intelligent learning system to enhance the quality of students' e-learning. Based on Taylor Principle and PDCA cycle theory, Wang and

Guo (2021) proposed a student-centered inquiry' Renewable Energy Sources (RES) curriculum model to foster their creativity and problem-solving skills.

However, little study has focused on how PDCA is integrated into the field of programming education, let alone investigating the effects of PDCA on reflective thinking in primary school students during tangible programming learning. According to the aforementioned discussion, programming can be used to develop reflective thinking, while the PDCA cycle can safeguard the quality of programming through reflective learning. We, therefore, assumed that students' programming skills and reflective thinking might improve if teachers apply the PDCA cycle strategy to programming instruction.

## 2.4 The revised PDCA model

In the present study, we tried to apply the PDCA cycle to improve students' programming skills and reflective thinking. As such, a revised model was proposed, as shown in Fig. 1. In this model, the process starts with P, representing drawing up a solution plan, followed by D - assembling the robot and programming it with programmable building blocks, and C - testing the results and debugging based on feedback. The model starts from P and passes through the D and C stages, showing improvement in multiple loops (single-loop, double-loop, triple-loop). Because tangible programming is aimed at younger children, debugging may not happen automatically, and careful and thoughtful educational guid-

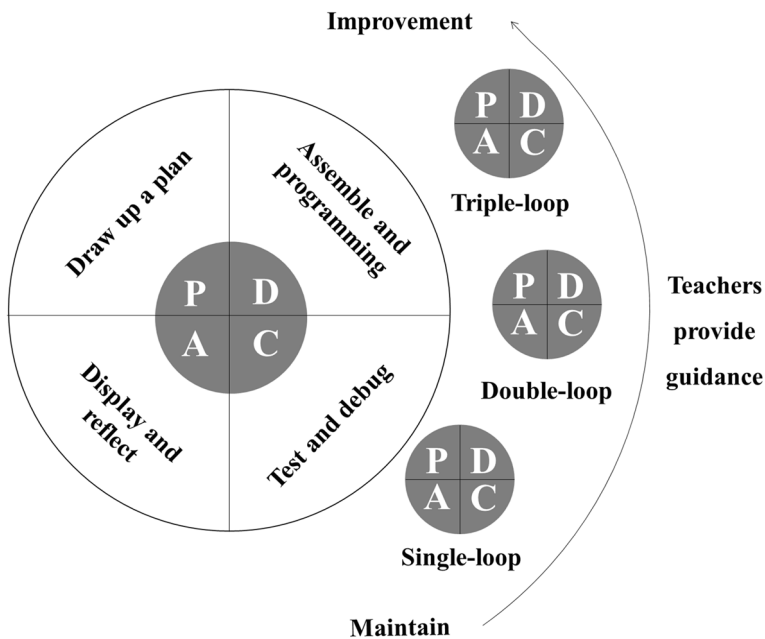


Fig. 1 The PDCA cycle model

ance is required (Gelter, 2003) for the quality of learning to be sustained in the right direction. Thus, we based on the multiple loops PDCA cycle, which both focuses on student learning and emphasizes the teachers' guidance role for children's tangible programming. Finally, the process concludes with A, showing and reflecting on the work.

## 2.5 Research questions

Based on the abovementioned theoretical background and empirical studies, the study aimed to explore the effectiveness of the PDCA cycle strategy in tangible programming learning by comparing it with the traditional teaching strategy. Student's reflective thinking levels and programming skills were examined to answer the following questions:

RQ1: What are the differences in the effectiveness of traditional teaching strategies and the PDCA cycle strategy in improving programming skills in the context of tangible programming?

RQ2: What are the differences in the effectiveness of traditional teaching strategies and the PDCA cycle strategy in improving reflective thinking levels in the context of tangible programming?

RQ3: What are the students' experiences of a course taught using the PDCA cycle strategy in the context of tangible programming?

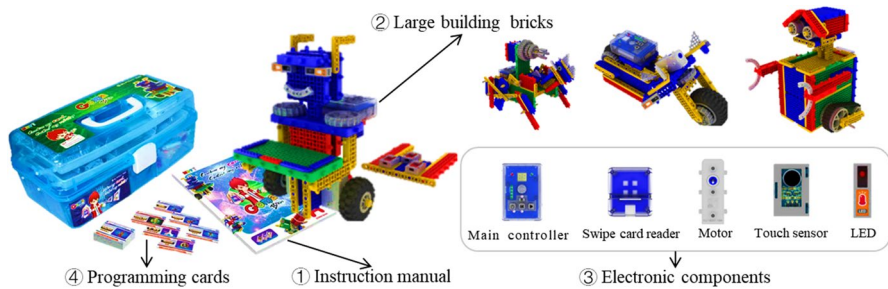
## 3 Methodology

### 3.1 Participants

A quasi-experimental design was adopted in this research, two classes with a total of 65 students were randomly selected from four second-grade classes in S primary school in Y city, Shandong province, China. Students were randomly dichotomized into the control ( $N=32$ ) and experiment ( $N=33$ ) groups. After the treatment, three students were eliminated from data analysis because two did not participate in the test, and another did not complete the test. As a result, the final count included only 62 students who were tested. Out of these, 31 students were in the experimental group (20 girls and 11 boys), and the remaining 31 were in the control group (20 girls and 11 boys). The mean age of the students was 8.63 years, with an age range between 7 and 9 years. All students explicitly volunteered to participate in the experiment and submitted their parents' consent.

### 3.2 Learning materials

The Magic Card Robot was chosen as the tangible programming aid for this study. Magic Card Robot aids were entry-level STEAM educational robotics kits explicitly designed for children aged 6–8 and widely used in young-adult tangible



**Fig. 2** Magic Card Robot aids

programming courses in most Asian countries, especially in Korea, China, and some southeast Asia countries. As shown in Fig. 2, Magic Card Robot aids consist of four parts: An instruction manual, Large building bricks, Electronic components (Main controller, Motor, Touch sensor, LED), and Programming cards. Children were encouraged to develop their programming skills using large building bricks and electronic components such as a main controller, motor, touch sensor, and LED. An instruction manual guided them to assemble various robot models. After completion, they could use a swipe card reader to scan programming cards, which were then scanned into the main board (as depicted in Fig. 3). By following this process, children could learn programming in a step-by-step manner and gain mastery of the three basic structures of sequential, repetitive, and conditional programming. These structures did not involve variables or operations and were designed to develop initial programming thinking and simple logical reasoning skills.






### 3.3 Experiment procedure

Before the course, one regular teacher in charge of tangible programming instruction and two research assistants who were responsible for addressing students' inquiries received five days of training to understand Magic Card Robot aids, the 'Magic Card Robot' course, and teaching methods (PDCA cycle strategy, traditional teaching strategy). The teaching experiment lasted eight weeks, with two sessions per week (one 90-minute session), and the research process consisted of three main phases (see Figs. 4 and 5).



**Fig. 3** Programming card scanning sequence



Task	Concrete content	Programming knowledge
<b>Task 1: Mechanical arm</b> 	<ol style="list-style-type: none"> <li>1. Understand the principle of gear transmission</li> <li>2. Add motor control</li> <li>3. Use simple choice and circular grammar structure to realize mechanical movement</li> </ol>	conditional structure repetitive structure
<b>Task 2: Service robot</b> 	<ol style="list-style-type: none"> <li>1. Make it clear that the wheel is a circular frame object around the shaft with the purpose of rotation</li> <li>2. Use motors, gears, etc., to build service robots</li> <li>3. Use simple selection and cycle syntax structure to realize the robot moving forward</li> </ol>	conditional structure repetitive structure
<b>Task 3: Mixing robot</b> 	<ol style="list-style-type: none"> <li>1. Understand the principle of the touch sensor (press manually to input numbered program commands)</li> <li>2. Use motors, gears, touch sensors, etc., to build agitators</li> <li>3. Use the sequence and circular grammar structure to realize the continuous rotation and stirring of food in one direction</li> </ol>	sequential structure repetitive structure
<b>Task 4: Obstacle avoidance robot</b> 	<ol style="list-style-type: none"> <li>1. Know how to touch the sensor to sense obstacles and realize obstacle avoidance function</li> <li>2. Use motors, gears, touch sensors, etc., to build obstacle avoidance robots</li> <li>3. Use sequence, selection, and cycle syntax structure to help bumper cars retreat and move forward after avoiding obstacles</li> </ol>	conditional structure repetitive structure sequential structure
<b>Task 5: Sweeping robot</b> 	<ol style="list-style-type: none"> <li>1. Use an infrared sensor to transmit/receive infrared, distinguish light and shade, and calculate the distance to avoid obstacles</li> <li>2. Use motors, gears, touch sensors, etc., to build a sweeping robot</li> <li>3. Use simple sequence, selection, and circular grammar structure to complete the cleaning function</li> </ol>	conditional structure repetitive structure sequential structure

**Fig. 4** Learning contents

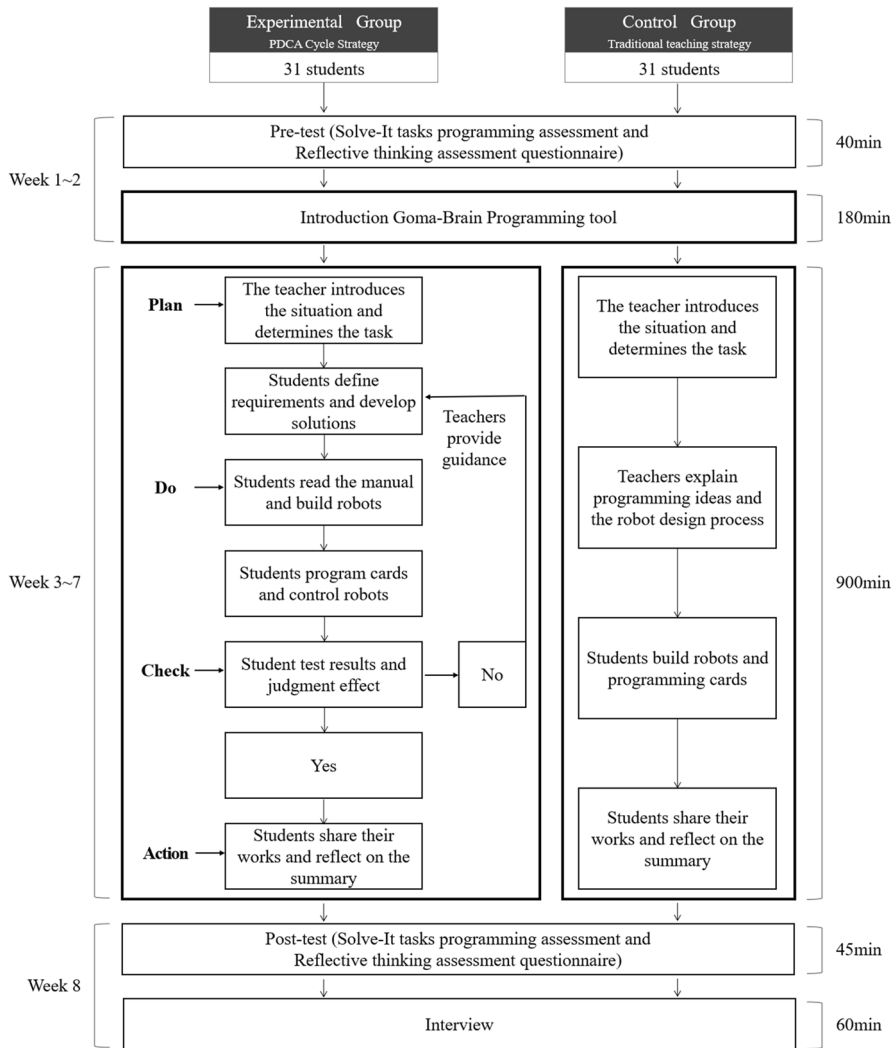
In the first phase (weeks 1–2), students' programming skills and reflective thinking levels were examined. Then, students were provided basic training on hardware related to Magic Card Robot aids, including the Main controller, Swipe card reader, Motor, Touch sensor, and LED.

The course was implemented in the second phase (weeks 3–7). The learning tasks were detailed in Fig. 4, which included a series of contents that integrate basic physic mechanical knowledge and sequential, repetitive, and conditional programming knowledge into real-world scientific applications. This course aimed to guide students in writing logical processes and displayed scientific phenomena using Magic Card Robot aids.

The experimental group used the PDCA cycle strategy, which consisted of four steps:

**P-Plan (draw up a plan)** The teacher introduced the problem situation and assigned the task. Students independently developed a solution (in the form of pseudo-code and flowchart) based on the task requirements;





**Fig. 5** Research procedure

**D-Do (assemble and programming)** Students independently built the robot according to the instruction manual and applied the card programming to manipulate the robot movement;

**C-Check (test and debug)** Students observed the movement of the programmed robot to judge whether the program input matched the expected output. When there was a mismatch, students debugged the existing errors under the guidance of the teacher (based on PDCA cycle iteration) to identify, analyze and correct the mistakes;

**A-Action (display and reflect)** Students independently analyzed the reasons for failure, summarized their experiences, and shared their successful works through the reflection iteration.

Subsequent three cycles were then built based on the experience from the previous cycle to enhance the quality of programming learning with continuous reflection and improvement. In the control group, a traditional teaching strategy was employed. Firstly, the teacher introduced the problem situation and determined the tasks to be completed, showing the general process of the solution. Next, the teacher taught the steps and points of building the robot, and students followed the teacher to apply the card programming to manipulate the robot movement. Finally, the students completed the work, and the teacher summarized the course.

In the third phase (week 8), a post-test was conducted to evaluate the student's programming skills and reflective thinking levels. Each student was asked to complete the test independently. Furthermore, six students were randomly selected for semi-structured interviews, each lasting 8–10 min, and recorded with the permission of the interviewees.

### 3.4 Instruments

This study was performed by using a mixed-methods approach. The quantitative data included the effectiveness of the application of the PDCA cycle strategy on students' programming skills and reflective thinking levels. These metrics were respectively measured using the "Solve-It Tasks Programming Assessment" (Sullivan & Bers, 2018) tool and the Reflective Thinking Assessment Questionnaire (Hong & Choi, 2019). For the qualitative data, semi-structured interviews were used to understand how students engaged in reflective learning and problem-solving during tangible programming.

#### 3.4.1 Solve-It Tasks Programming Assessment

The Solve-It Tasks Programming Assessment (Solve-It Tasks) was developed by the DevTech research group at Tufts University (Sullivan & Bers, 2018) to examine young children's knowledge of foundational programming concepts ranging from sequencing, repetitive and conditional structures (Strawhacker & Bers, 2015). The assessment consisted of five parts: Easy Sequencing, Hard Sequencing, Easy repetitive, Hard repetitive, and Using the conditional.

This study utilized a series of "Solve-It Tasks" to assess students' programming skills. Solve-It 1 and 2 assessed students' mastery of sequential structure knowledge, which required students to write instructions for robot movements in a specific sequence. Solve-It 3 and 4 focused on the repetitive structure, which required students to incorporate instructions into more extended programs and create subroutines. Solve-It 5 tested conditional structure by asking students to write instructions for the robot to make choices based on specific conditions. Relatively speaking, Solve-It 1 and 3 assessed basic programming concepts, while

Solve-It 2, 4, and 5 evaluated more complex programming skills. The difference between simple and complex tasks for sequential and repetitive structures was determined by the number of cards needed to order the program correctly.

The test was conducted in week 8, where students were asked to listen to a story about robots based on their familiar background and then try to create a program using programming cards. The “Dinosaur” test case is shown in Fig. 6. After the teacher finished reading the story once, students were asked to arrange the program cards according to the story, and then the teacher repeated the story for students to check and modify their programs.

The scoring rules for “Solve-It Tasks” were based on a two-stage scoring system (Sullivan & Bers, 2018), including the position of ‘Begin’ and ‘End’ (0 to 3 points) and the relative order of the action blocks (0 to 3 points). Each question was scored on a scale of 0–6 based on the correctness of the student’s.

program, with a total of 30 points. The scoring was done by two research assistants, and after the scoring was completed, they exchanged and reviewed each other’s work. If there was a discrepancy, they discussed it and came to a consensus on the score. The Cronbach’s Alpha coefficient value of the Solve-It Tasks Programming Assessment was 0.878, indicating a high level of reliability in the test results.

### 3.4.2 Reflective Thinking Assessment Questionnaire

The Reflective Thinking Assessment Questionnaire was developed based on the scale of “Assessing Reflective Thinking in Solving Design Problems(ARTi D)”(Hong & Choi, 2019). The finalized scale included three dimensions, which were single-loop reflection, double-loop reflection, and triple-loop reflection, as shown in Table 1.

Single-loop reflection involved reflecting on the effectiveness of actions taken in order to achieve predetermined goals, focusing on identifying and correcting errors. In contrast, double-loop reflection went beyond this by also questioning the underlying assumptions and goals themselves and generating new goals that may better align with desired outcomes (Flood & Romm, 1996). This iterative process

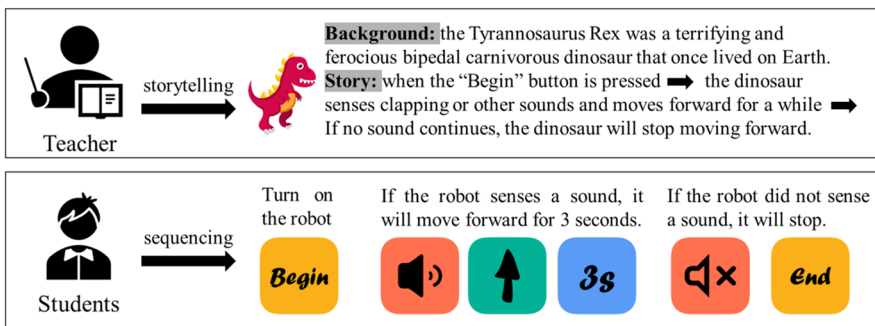


Fig. 6 “Dinosaur” test case

**Table 1** Reflective Thinking Assessment Questionnaire

Dimensions	Item
Single-loop	I evaluated whether the sequencing could be done effectively.
	I checked whether the sequencing scheme could successfully get the robot to move.
	I evaluated whether the sequencing allowed the robot to move correctly.
Double-loop	I re-examined the sequencing scheme to identify existing problems.
	I re-judged my previous understanding of the problem.
	I checked why the new sequencing was critical to solving the problem.
Triple-loop	I assessed that my sequencing was compliant.
	I assessed that the new sequencing was the best solution.
	I found ways or laws to get the robot moving.
	I consider the ethical issues associated with tangible programming.

was similar to the PDCA cycle strategy. Finally, triple-loop reflection involved critical reflection (Mezirow, 1990) on the presuppositions and values underlying one's programming process, as well as the larger social and educational context in which it occurred, intending to create more just and equitable outcomes.

The Reflective Thinking Assessment Questionnaire consisted of 10 questions on a 5-point Likert rating (1 = never, 2 = seldom, 3 = sometimes, 4 = often, 5 = always). The total score on the scale was 50 points. The total score indicated the level of reflective thinking, and the questionnaire demonstrated high internal consistency with a Cronbach's Alpha value of 0.803.

### 3.4.3 Semi-structured interviews

At the end of the course, a semi-structured interview was conducted among six students randomly selected from the experimental group to understand their perceptions and experiences of the programming class supported by the PDCA cycle. We aimed to provide some qualitative evidence to support the quantitative results. The interview questions were designed as follows.

Question 1: Do you enjoy the 'Magic Card Robot' course? What do you learn from this course?

Question 2: Have you encountered any problems in the process of tangible programming? If so, how did you solve it?

Question 3: Which part of the class do you think attracts you most?

## 4 Results

### 4.1 Programming skills

Students' programming skills were reflected by their "Solve-It Task" performance, consisting of three sub-dimensions (i.e., sequential, repetitive, conditional

**Table 2** The paired-sample *t*-test result of the overall programming skills

Group	Pairs	Paired Differences				<i>t</i>	Probability (two-tailed)	Cohen's <i>d</i>
		Mean	SD	95% CI of the difference				
				Lower	Upper			
Experimental	Pre-test & Post-test	-11.160	2.853	-12.208	-10.115	-21.782	0.000**	3.912
Control	Pre-test & Post-test	-6.520	4.265	-8.081	-4.952	-8.506	0.000**	1.528

\*\* $p < .01$ . CI=Confidence Interval

**Table 3** The paired-sample *t*-test on the two groups' three sub-dimensions of the programming skills

Variables	Group	Pairs	Paired Differences				<i>t</i>	Probabil- ity (two- tailed)	Cohen's <i>d</i>
			Mean	SD	95% CI of the difference				
					Lower	Upper			
Sequential	Experimental	Pre-test & Post-test	-2.400	0.724	-2.669	-2.138	-18.491	0.000**	3.321
	Control		-1.630	1.231	-2.081	-1.177	-7.366	0.000**	1.323
Repetitive	Experimental	Pre-test & Post-test	-2.190	0.873	-2.514	-1.874	-13.998	0.000**	2.514
	Control		-1.030	1.103	-1.437	-0.628	-5.213	0.000**	0.936
Conditional	Experimental	Pre-test & Post-test	-1.970	1.329	-2.455	-1.480	-8.245	0.000**	1.486
	Control		-1.190	1.515	-1.749	-0.638	-4.387	0.000**	0.788

\*\* $p < .01$ . CI=Confidence Interval

structure) of five tasks. To answer question 1, we investigated students' improvement of programming skills over the experiment and compared the differences between the two groups. Independent sample *t*-test showed that, in terms of students' prior programming skills, there was no significant difference ( $t = 0.684$ ,  $p > .05$ ) between the experimental group ( $MD = 12.936$ ,  $SD = 2.632$ ) and the control group ( $MD = 12.484$ ,  $SD = 2.567$ ). Furthermore, students in both groups scored approximately 13 out of 30 points, indicating ample room for improvement in their programming skills.

We then compared their improvement in programming skills. Table 2 showed that students in both groups significantly improved (experiment group:  $p < .05$ , Cohen's  $d = 3.912$ ; control group:  $p < .05$ , Cohen's  $d = 1.528$ ).

We subsequently analyzed the differences between the three sub-dimensions. As shown in Table 3, the paired-sample *t*-tests revealed significant improvements in the three sub-dimensions of programming skills for both groups. Specifically, the *p*-value for the sequential structure to conditional structure tasks were 0.000, 0.000, and 0.000 in experiment groups.000, 0.000, 0.000 in control group. In general,

**Table 4** The one-way ANCOVA result of the overall programming skills

Variables	Group	N	Mean	SD	Adjusted mean	Adjusted SE	F	Partial $\eta^2$
Programming skills	Experimental	31	24.100	2.561	24.065	0.519	46.851***	0.443
	Control	31	19.000	3.173	19.032			

\*\*\* $p < .001$ **Table 5** The one-way ANCOVA on the two groups' three sub-dimensions of the programming skills

Variables	Group	N	Mean	SD	Adjusted mean	Adjusted SE	F	Partial $\eta^2$
Sequential	Experimental	31	5.177	0.599	5.166	0.116	32.908***	0.358
	Control	31	4.210	0.693	4.221			
Repetitive	Experimental	31	4.758	0.644	4.760	0.127	37.363***	0.388
	Control	31	3.661	0.757	3.660			
Conditional	Experimental	31	4.226	1.117	4.218	0.203	10.931**	0.156
	Control	31	3.258	1.125	3.266			

\*\*\* $p < .001$ , \*\* $p < .01$ . SD = standard deviation; SE = standard error

tangible programming has effectively developed primary school children's programming skills and three sub-dimensions of programming skills.

To compare the differences between the two groups' Solve-It Tasks performance, a one-way analysis of covariance (ANCOVA) was conducted by using the pre-test scores as the covariate and experimental conditions as independent variables. The F-test results for the product terms of experimental conditions and pre-test Programming skills did not violate the homogeneity-of-slopes assumption ( $F = 3.186$ ,  $p > .05$ ), indicating it was sensible to perform the ANCOVA test.

Table 4 shows that, in terms of programming skills, the students in the experimental group performed significantly better than those in the control group ( $F = 46.851$ ,  $p < .001$ , partial  $\eta^2 = 0.443$ ). The results demonstrated that using the PDCA cycle strategy more effectively developed students' programming skills than the traditional teaching method.

After confirming the effect of the PDCA strategy on students' overall programming skills, we compared its three sub-dimensions (see Table 5). Specifically, the experimental group had a clear understanding of the sequential structure ( $MD = 5.177$ ) and could sequence the algorithmic instructions accurately. The experimental group also scored higher on the repetitive structure ( $MD = 4.758$ ), showing that they could create simple repetitive programs, i.e., embedding repetitive instructions in a series of program instructions and ensuring the integrity of the instructions by repeating a core set of steps. The conditional structure required linking specific actions to the conditions that make them necessary or desirable, among which the experimental group performed a conditional test score of ( $MD = 4.226$ ), meaning that they were capable of mastering basic logical reasoning.

**Table 6** The one-way ANCOVA on the two groups' reflective thinking levels survey

Variables	Group	N	Mean	SD	Adjusted mean	Adjusted SE	F	Partial $\eta^2$
Reflective thinking levels	Experimental	31	43.169	3.567	43.156	0.812	39.376***	0.400
	Control	31	35.936	5.385	35.949			

\*\*\* $p < .001$ **Table 7** Distribution rates of students' reflective thinking levels in pre-test/post-test

Dimensions	Experimental (%)		Control (%)	
	Pre-test	Post-test	Pre-test	Post-test
Single-loop	54.84	16.13	64.52	51.61
Double-loop	32.26	58.06	19.35	29.03
Triple-loop	12.90	25.81	16.13	19.35

the distribution rates refer to the proportion of students with these reflective thinking levels (single-loop, double-loop, three-loop) in the total number of students in this class

## 4.2 Reflective thinking levels

To explore students' reflective thinking levels, we assigned the dimension where students scored highest as their individual reflective thinking levels. The results showed that prior to the intervention, out of 62 students, 61.29% ( $n=38$ ) students were at the single-loop reflection level, 24.19% ( $n=15$ ) students hit the double-loop reflection level, and 14.52% ( $n=9$ ) students reached the triple-loop reflection level. Moreover, students presented comparable levels in both groups ( $t=0.121$ ,  $p>.05$ ). The results indicated that their entry behavior was similar. Therefore, we continued to do the following analysis.

The effect of the PDCA cycle strategy on reflective thinking levels was further assessed. A one-way ANCOVA analysis was conducted by using the pre-test reflective thinking levels as a covariate, the post-test reflective thinking level as an independent variable, and experimental conditions as a dependent variable. The F-test results for the pre-test and post-test reflective thinking levels and experimental conditions did not violate the homogeneity-of-slopes assumption ( $F=2.947$ ,  $p>.05$ ), indicating it was sensible to perform the ANCOVA test. It was found that students in the experimental group achieved significantly higher levels of reflective thinking than those in the control group ( $F=39.376$ ,  $p<.001$ , partial  $\eta^2=0.036$ ), as shown in Table 6.

Table 7 showed the students' ARTiD's reflective thinking levels before and after the experiment. For the experimental group, 58.06% ( $n=18$ ) achieved double-loop reflection, demonstrating that students who intervened by the PDCA cycle strategy were enabled to revisit continually, reasonably question, and iterate on their already defined goals or programs to achieve better learning outcomes. For students in the control group, 51.61% ( $n=16$ ) were in single-loop reflection, where they sought to match problems quickly and think linearly, completing tangible programming



tasks. Therefore, the PDCA cycle strategy effectively improved students' reflective thinking.

### 4.3 Students' experience of PDCA cycle strategy and tangible programming learning

To investigate the effect of PDCA strategy on students' experience in a tangible programming class. A semi-structured interview was subsequently conducted among six students, who were randomly selected in the experimental group. "Their experience with the 'Magic Card Robot' course, the problems they encountered during the course and how they solved them, and their favorite part of the course" were interviewed.

Firstly, the experience of the 'Magic Card Robot' course. All six students said they enjoyed the course and the learning style. For instance, student A indicated that the 'Magic Card Robot' course combined the knowledge needed to learn with practical applications and made me enjoy exploring the world of science. Student B referred, "I loved the robots. In this way of learning, I have used my imagination to put together many different shapes of robots and then utilized the swipe card programming function to make the robots move, which was very helpful for my programming learning."

Secondly, the problems encountered during the learning process and the solutions. When students were asked about the difficulties they encountered in the process of tangible programming, most students reported that on the first attempt, the robot usually did not move forward as required. For example, student C said, "I could not succeed on the first attempt, but instead needed to keep checking and adjusting to get the robot to move." Programming skills are spiraling, requiring most students to go through multiple PDCA cycles and improve their programming skills. In addition, Student E expressed, "When I had trouble programming, I sought help from my teacher. The teacher carefully explained the meaning of each step, which helped me achieve my goal." It also reflects the necessity of providing targeted guidance during the testing and debugging stage.

Finally, students' favorite part of the course. The final display and reflect session were favored by the students. Student F said, "Whenever the class ended, everyone presented their work individually and showed how to use repetitive structures to make the robot program easy, which I will try to do in future classes." It is clear from this process that the PDCA cycle strategy prompts students to reflect more profoundly to standardize and replicate successful experiences.

## 5 Discussion

Programming skills are measured by the results of the programming tasks, which mainly consist of sequential, conditional, and repetitive structure, as a way to examine students' hands-on programming operational skills(Kuo & Hsu, 2020). The

results showed that the programming skills of both groups improved significantly, indicating that the students understood the concepts related to sequential, conditional, and repetitive structure and had the programming practical operation ability. This finding was in line with the findings of Sullivan and Bers's (2016) and Cejka et al. (2006) studies. Furthermore, a comparison between groups revealed that the experimental group performed better on the programming skills test and scored higher on the task. In-depth analyzes of the reasons for the more significant improvement of programming skills in the experimental group lay in the fact that the PDCA cycle strategy has the same logic as solving programming problems (Daminda Kuruppu, 2022). It also helped students understand and reflect on the problems encountered in their learning, encouraged them to compare and analyze the gap between achieved and expected goals, and discussed current shortcomings and strategies for improvement. Students in the experimental group were guided by the PDCA cycle strategy to improve their programming skills by understanding problems, reflecting on improvements, and solving problems (Choo et al., 2007). This finding reaffirmed the positive effect of tangible programming aids.

In terms of reflective thinking level, students in the experimental group performed better than those in the control group, students in the experimental group mostly presented a double-loop reflection level. One possible reason is that every student was required to test and debug (The "Check" in PDCA) through an iterating process. In this case, they were more prone to critically examine the process and results of their investigations and repeatedly adjust the programming sequence to see if the problem was addressed correctly (Flood & Romm, 1996; Schepers & Wetzels, 2007). This finding also tallies with Staudinger (2013)'s study, which found that reflection may lead to self-insight. The control group students were instructed to use flowcharts to imitate the teacher's behavior until they achieved the goal. As a result, the majority of them exhibited single-loop reflection, indicating linear problem-solving skills, which is consistent with the findings of Adams et al. (2003). In a nutshell, our study confirmed that a PDCA cycle strategy involving iterative reflection is an effective way to facilitate students' learning from surface to depth in tangible programming (M. Wang et al., 2018), which has practical implications for educators to be able to select better and implement instructional strategies.

Some qualitative data were used to analyze students' experience of the tangible programming process. The results showed that students presented a positive perception of learning tangible programming. This corresponds to the study of Sapounidis et al. (2015) that students who use physical programming aids have a significant advantage in terms of motivation to learn. Moreover, the results of the interviews confirmed the conclusion that students can improve their programming skills by solving programming problems. For example, one student reported utilizing pseudo-code and flowcharts to structure and systematize the assembly and programming of the robot. This may be because he can clearly plan programming operations, make abstract programming concepts gradually and concretely, and finally improve his programming skills. This speculation is consistent with the argument of Morgan and Stewart (2017). And, students emphasized that they liked the reflection session the most. As one student mentioned in the interview, he had transformed his fear of failure into a positive attitude towards continually trying and improving, increasing

his awareness of programming challenges. This finding supports the conclusions of (Widmer et al., 2009), who posited that students who engaged in regular reflection were more attuned to changes in their surroundings and the outcomes of their actions. To sum up, our study confirmed that the PDCA cycle strategy positively impacts the enhancement of programming skills and reflective thinking in primary school students' tangible programming learning.

## 6 Conclusion and limitation

### 6.1 Conclusion

In this study, we aimed to explore the impact of the PDCA cycle strategy on children's programming skills and reflective thinking levels in the 'Magic Card Robot' course. The results showed that tangible programming facilitated students' programming skills development and the three dimensions of programming skills (sequential structure, conditional structure, repetitive structure). Further, compared with traditional teaching strategies, applying the PDCA cycle strategy in tangible programming courses can achieve better learning outcomes. Specifically, PDCA is a four-in-one mode of "(P)draw up a plan, (D)assemble and programming, (C)test and debug, display and reflect (A)" to create a closed loop of education. One of the critical links in the PDCA cycle chain is the test and debug, which is also the catalyst for the next cycle. The findings confirm that PDCA can promote the improvement of children's reflective thinking, which extends and strengthens prior research on developing reflective thinking levels in tangible programming. The findings can be served as a teaching case about implementing tangible programming for primary school teachers.

### 6.2 Limitation and future direction

The following research limitations of this research should be noted. Firstly, the study only involved students in grade 2, and the sample size was not rich enough, which may cause bias in the results. In the future, it is necessary to extend the primary student group to the kindergarten group and demonstrate the application effect of the PDCA cycle strategy in a tangible programming course. Secondly, the study did not consider students' accidental success in solving programming problems through trial and error. Therefore, in the future, it is necessary to add scaffolding to the PDCA cycle strategy to guide debugging practices (Chiu & Huang, 2015). Through a carefully designed failure task (Kapur, 2008), the teacher guides the students to reflect on what led to the error in the failed task. Students are asked to try implementing a programming solution to this error to reflect on and fix all mistakes to avoid repeating them. What is more, video coding technology can be used to record students' actual actions to analyze children's behavior patterns more accurately in the future.

**Authors' contributions** Xin Gong: Conceptualization, Methodology, Data curation, Formal analysis, Project administration, Writing – original draft, Writing – review & editing. Shufan Yu: Conceptualization, Methodology, Writing – review & editing. Jie Xu: Formal analysis, Writing – review & editing. Ailing Qiao: Funding acquisition, Supervision, Resources, Writing – review & editing. Han Han: Investigation.

**Funding** This work was supported by the 2022 Key research project of the Chinese Ministry of Education (DCA220449) and the Beijing Education Science Plan 2021 Key Project (CDAA 21048).

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

## Declarations

**Ethics approval** All procedures performed in the study involving human participants were in accordance with the World Medical Association Declaration of Helsinki. The research participants and their parents agreed to participate in the study and their complete anonymity was ensured.

**Competing interests** The authors declare that they have no conflict of interest.

## References

- Ackermann, E. K. (1996). Perspective-taking and object construction: Two keys to learning. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice* (pp. 1, 25–37). Lawrence Erlbaum Associates, Inc.
- Adams, R. S., Turns, J., & Atman, C. J. (2003). Educating effective engineering designers: The role of reflective practice. *Design Studies*, 24(3), 275–294. [https://doi.org/10.1016/S0142-694X\(02\)00056-X](https://doi.org/10.1016/S0142-694X(02)00056-X)
- Alonso, J. M. (2020). Teaching explainable Artificial Intelligence to High School Students. *International Journal of Computational Intelligence Systems*, 13(1), 974. <https://doi.org/10.2991/ijcis.d.200715.003>
- Altın, M., & Saracaloğlu, A. S. (2018). Creative, critical and reflective thinking: Similarities-differences. *Uluslararası Güncel Eğitim Araştırmaları Dergisi (UGEAD)*, 4(1), 1–9.
- 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, 105954. <https://doi.org/10.1016/j.chb.2019.03.018>
- Antonio, R. P. (2020). Developing students' reflective thinking skills in a Metacognitive and Argument-Driven Learning Environment. *International Journal of Research in Education and Science*, 6(3), 467–483. <https://doi.org/10.46328/ijres.v6i3.1096>
- Bayrak, F., & Usluel, Y. K. (2011). The effect of blogging on reflective thinking skill. *Hacettepe University Journal of Education*, 40, 93–104.
- Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. *High-Tech Tots: Childhood in a Digital World*, 49, 49–70.
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72, 145–157. <https://doi.org/10.1016/j.compedu.2013.10.020>
- Bers, M. U., González-González, C., & Armas-Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education*, 138, 130–145. <https://doi.org/10.1016/j.compedu.2019.04.013>
- Blagojević, M., & Micić, Ž. (2013). A web-based intelligent report e-learning system using data mining techniques. *Computers & Electrical Engineering*, 39(2), 465–474. <https://doi.org/10.1016/j.compeleceng.2012.09.011>
- Burleson, W. S., Harlow, D. B., Nilsen, K. J., Perlin, K., Freed, N., Jensen, C. N., Lahey, B., Lu, P., & Muldner, K. (2018). Active learning environments with robotic tangibles: Children's physical and

- virtual spatial programming experiences. *IEEE Transactions on Learning Technologies*, 11(1), 96–106. <https://doi.org/10.1109/TLT.2017.2724031>
- Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education*, 22(4), 711–722.
- Chen, H. M., Nguyen, B. A., Yan, Y. X., & Dow, C. R. (2020). Analysis of learning behavior in an Automated Programming Assessment Environment: A Code Quality Perspective. *IEEE Access: Practical Innovations, Open Solutions*, 8, 167341–167354. <https://doi.org/10.1109/ACCESS.2020.3024102>
- Chen, H. E., Sun, D., Hsu, T. C., Yang, Y., & Sun, J. (2023). Visualising trends in computational thinking research from 2012 to 2021: A bibliometric analysis. *Thinking Skills and Creativity*, 47, 101224. <https://doi.org/10.1016/j.tsc.2022.101224>
- Chiu, C. F., & Huang, H. Y. (2015). Guided Debugging Practices of Game based programming for novice programmers. *International Journal of Information and Education Technology*, 5(5), 343–347. <https://doi.org/10.7763/IJiet.2015.V5.527>
- Cho, Y., & Lee, Y. (2017). Possibility of improving computational thinking through activity based learning. *Journal of Theoretical and Applied Information Technology*, 95(18), 4385–4393. <https://www.jatit.org/volumes/Vol95No18/6Vol95No18.pdf>. Accessed 14 Jan 2023.
- Choo, A. S., Linderman, K. W., & Schroeder, R. G. (2007). Method and context perspectives on learning and knowledge creation in quality management. *Journal of Operations Management*, 25(4), 918–931. <https://doi.org/10.1016/j.jom.2006.08.002>
- Daminda Kuruppu, K. A. D. (2022). Education Reform as a platform to improve interactions of the Engineering Students during Online Teaching and learning at higher education amidst Covid-19 pandemic. *International Journal of Educational Reform*, 31(2), 202–217. <https://doi.org/10.1177/10567879211042327>
- Deming, W. E. (2000). *The new economics, for industry, government, education* (2nd ed.). MIT Press.
- Demir, F. (2022). The effect of different usage of the educational programming language in programming education on the programming anxiety and achievement. *Education and Information Technologies*, 27(3), 4171–4194. <https://doi.org/10.1007/s10639-021-10750-6>
- Evripidou, S., Amanatiadis, A., Christodoulou, K., & Chatzichristofis, A. (2021). Introducing algorithmic thinking and sequencing using tangible Robots. *IEEE Transactions on Learning Technologies*, 14(1), 93–105. <https://doi.org/10.1109/TLT.2021.3058060>
- Feijoo-Almonacid, A., & Rodriguez-Garavito, C. H. (2022). Hardware-Software platform for the development of STEM skills. *IEEE Revista Iberoamericana de Tecnologías Del Aprendizaje*, 17(2), 170–177. <https://doi.org/10.1109/RITA.2022.3166969>
- Fischer, T., & Lau, W. (2006, June). Marble track music sequencers for children. *Proceedings of the 2006 Conference on Interaction Design and Children*, (pp.141–144). <https://doi.org/10.1145/1139073.1139108>
- Flood, R. L., & Romm, N. R. A. (1996). Plurality revisited: Diversity management and triple loop learning. *Systems Practice*, 9(6), 587–603. <https://doi.org/10.1007/BF02169215>
- Gelter, H. (2003). Why is reflective thinking uncommon. *Reflective Practice*, 4(3), 337–344. <https://doi.org/10.1080/1462394032000112237>
- Havenga, M., Breed, B., Mentz, E., Govender, D., Govender, I., Dignum, F., & Dignum, V. (2013). Metacognitive and problem-solving skills to Promote Self-Directed Learning in Computer Programming: Teachers' Experiences. *SA-eDUC Journal*, 10(2), 1–14.
- Hellberg, R., & Fauskanger, E. (2022). Learning of quality improvement theory – experiences with reflective learning from a student perspective. *International Journal of Lean Six Sigma*. <https://doi.org/10.1108/IJLSS-04-2022-0090>
- Hong, Y. C., & Choi, I. (2019). Relationship between student designers' reflective thinking and their design performance in bioengineering project: Exploring reflection patterns between high and low performers. *Educational Technology Research and Development*, 67(2), 337–360. <https://doi.org/10.1007/s11423-018-9618-6>
- Hsieh, M. C., Pan, H. C., Hsieh, S. W., Hsu, M. J., & Chou, S. W. (2022). Teaching the Concept of Computational thinking: A STEM-Based program with tangible Robots on Project-Based learning courses. *Frontiers in Psychology*, 12, 828568. <https://doi.org/10.3389/fpsyg.2021.828568>
- 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 & Education*, 126, 296–310. <https://doi.org/10.1016/j.compedu.2018.07.004>

- Iivari, N., Sharma, S., & Ventä-Olkkonen, L. (2020). Digital transformation of everyday life – how COVID-19 pandemic transformed the basic education of the young generation and why information management research should care? *International Journal of Information Management*, 55, 102183. <https://doi.org/10.1016/j.ijinfomgt.2020.102183>
- Kapur, M. (2008). Productive failure. *Cognition and Instruction*, 26(3), 379–424. <https://doi.org/10.1080/07370000802212669>
- Kizilkaya, G., & Aşkar, P. (2009). The development of a reflective thinking skill scale towards problem solving. *Eğitim ve Bilim*, 34(154), 82–92.
- Kuo, W. C., & Hsu, T. C. (2020). Learning computational thinking without a computer: How computational participation happens in a computational thinking Board game. *The Asia-Pacific Education Researcher*, 29(1), 67–83. <https://doi.org/10.1007/s40299-019-00479-9>
- Lai, X., & Wong, G. K. (2022). Collaborative versus individual problem solving in computational thinking through programming: A meta-analysis. *British Journal of Educational Technology*, 53(1), 150–170. <https://doi.org/10.1111/bjet.13157>
- Liao, H. C., & Wang, Y. H. (2019). Reflective thinking scale for Healthcare Students and Providers—Chinese version. *Social Behavior and Personality: An International Journal*, 47(2), 1–10. <https://doi.org/10.2224/sbp.7671>
- Lin, Y. T., Yeh, M. K. C., & Tan, S. R. (2022). Teaching programming by revealing thinking process: Watching experts' live coding videos with reflection annotations. *IEEE Transactions on Education*, 65(4), 617–627. <https://doi.org/10.1109/TE.2022.3155884>
- Malik, S. I., Tawafak, R. M., & Shakir, M. (2021). Aligning and assessing teaching approaches with SOLO taxonomy in a computer programming course. *International Journal of Information and Communication Technology Education*, 17(4), 1–15. <https://doi.org/10.4018/IJICTE.20211001.oa5>
- Marsh, J., Plowman, L., Yamada-Rice, D., Bishop, J., & Scott, F. (2016). Digital play: A new classification. *Early Years*, 36(3), 242–253. <https://doi.org/10.1080/09575146.2016.1167675>
- Marshall, P. (2007). Do tangible interfaces enhance learning? *Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI' 07)*, 163–170. <https://doi.org/10.1145/1226969.1227004>
- Mezirow, J. (1990). *Fostering critical reflection in adulthood* (pp. 1–20). Jossey-Bass Publishers.
- Morgan, S. D., & Stewart, A. C. (2017). Continuous improvement of Team assignments: Using a web-based Tool and the Plan-Do-Check-act cycle in design and redesign: Continuous improvement of Team assignments. *Decision Sciences Journal of Innovative Education*, 15(3), 303–324. <https://doi.org/10.1111/dsji.12132>
- Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children (MIT AI lab memo no. 360/Logo memo, N. 24) [MIT AI Lab]. <https://hdl.handle.net/1721.1/5784>
- Piaget, J. (1959). *The language and thought of the child* (3d ed). Humanities Press.
- Piaget, J. (1973). *The child and reality: Problems of genetic psychology*. Grossman.
- Revelle, G., Zuckerman, O., Druin, A., & Bolas, M. (2005). Tangible user interfaces for children. *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, 2051–2052. <https://doi.org/10.1145/1056808.1057095>
- Salido, A., & Dasari, D. (2019). The analysis of students' reflective thinking ability viewed by students' mathematical ability at senior high school. *Journal of Physics: Conference Series*, 1157(2), 022121. <https://doi.org/10.1088/1742-6596/1157/2/022121>
- Sapounidis, T., Demetriadis, S., & Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing*, 19(1), 225–237. <https://doi.org/10.1007/s00779-014-0774-3>
- Sapounidis, T., Stamovlasis, D., & Demetriadis, S. (2019). Latent class modeling of children's preference profiles on tangible and graphical Robot Programming. *IEEE Transactions on Education*, 62(2), 127–133. <https://doi.org/10.1109/TE.2018.2876363>
- Schepers, J., & Wetzels, M. (2007). A meta-analysis of the technology acceptance model: Investigating subjective norm and moderation effects. *Information & Management*, 44(1), 90–103. <https://doi.org/10.1016/j.im.2006.10.007>
- Schweikardt, E., & Gross, M. D. (2008). The robot is the program: Interacting with roBlocks. *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction* (pp.167–168). <https://doi.org/10.1145/1347390.1347427>
- Shanley, N., Martin, F., Hite, N., Perez-Quinones, M., Ahlgrim-Delzell, L., Pugalee, D., & Hart, E. (2022). Teaching programming online: Design, Facilitation and Assessment Strategies and Recommendations for High School Teachers. *TechTrends*, 66(3), 483–494. <https://doi.org/10.1007/s11528-022-00724-x>



- Shewhart, W. A. (1931). *Economic control of quality of manufactured product*. D. Van Nostrand.
- Shim, J., Kwon, D., & Lee, W. (2017). The Effects of a Robot Game Environment on Computer Programming Education for Elementary School Students. *IEEE Transactions on Education*, 60(2), 164–172. <https://doi.org/10.1109/TE.2016.2622227>
- Silvis, D., Lee, V. R., Clarke-Midura, J., & Shumway, J. F. (2022). The technical matters: Young children debugging (with) tangible coding toys. *Information and Learning Sciences*, 123(9/10), 577–600. <https://doi.org/10.1108/ILS-12-2021-0109>
- Smith, W. (2009). Theatre of Use: A Frame analysis of Information Technology demonstrations. *Social Studies of Science*, 39(3), 449–480. <https://doi.org/10.1177/0306312708101978>
- Socratous, C., & Ioannou, A. (2020). Common errors, successful debugging, and engagement during block-based programming using educational robotics in elementary education. *14th International Conference of the Learning Sciences*, 2, 991–998. <https://doi.org/10.22318/icls2020.991>
- Staudinger, U. M. (2013). The need to distinguish personal from general wisdom: a short history and empirical evidence. In M. Ferrari & N. M. Weststrate (Eds.), *The Scientific Study of Personal Wisdom* (pp.3–19). Springer Netherlands. [https://doi.org/10.1007/978-94-007-7987-7\\_1](https://doi.org/10.1007/978-94-007-7987-7_1)
- Strawhacker, A., & Bers, M. U. (2015). I want my robot to look for food”: Comparing Kindergarten’s programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, 25(3), 293–319. <https://doi.org/10.1007/s10798-014-9287-7>
- Su, Y. S., Shao, M., & Zhao, L. (2022). Effect of mind mapping on creative thinking of children in scratch visual programming education. *Journal of Educational Computing Research*, 60(4), 906–929. <https://doi.org/10.1177/07356331211053383>
- Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, 26(1), 3–20. <https://doi.org/10.1007/s10798-015-9304-5>
- Sullivan, A., & Bers, M. U. (2018). Dancing robots: Integrating art, music, and robotics in Singapore’s early childhood centers. *International Journal of Technology and Design Education*, 28(2), 325–346. <https://doi.org/10.1007/s10798-017-9397-0>
- Sun, C., Shute, V. J., Stewart, A. E. B., Beck-White, Q., Reinhardt, C. R., Zhou, G., Duran, N., & D’Mello, S. K. (2022). The relationship between collaborative problem solving behaviors and solution outcomes in a game-based learning environment. *Computers in Human Behavior*, 128, 107120. <https://doi.org/10.1016/j.chb.2021.107120>
- Tan, S. Y. (2021). Reflective learning? Understanding the student perspective in higher education. *Educational Research*, 63(2), 229–243. <https://doi.org/10.1080/00131881.2021.1917303>
- Walasek, T. A., Kucharczyk, Z., & Morawska-Walasek, D. (2011). Assuring quality of an e-learning project through the PDCA approach. *Archives of Materials Science and Engineering*, 48(1), 56–61.
- Walther, J., Sochacka, N. W., & Kellam, N. N. (2011). Emotional Indicators as a Way to Initiate Student Reflection in Engineering Programs. In *2011 ASEE Annual Conference & Exposition Proceedings* (pp.22–557). <https://doi.org/10.18260/1-2--17838>
- Wang, X., & Guo, L. (2021). How to promote University students to innovative use renewable energy? An Inquiry-Based Learning Course Model. *Sustainability*, 13(3), 1418. <https://doi.org/10.3390/su13031418>
- Wang, M., Yuan, B., Kirschner, P. A., Kushniruk, A. W., & Peng, J. (2018). Reflective learning with complex problems in a visualization-based learning environment with expert support. *Computers in Human Behavior*, 87, 406–415. <https://doi.org/10.1016/j.chb.2018.01.025>
- Widmer, P. S., Schippers, M. C., & West, M. A. (2009). Recent developments in reflexivity research: A review. *Psychology of Everyday Activity*, 2(2), 2–11.
- Wyeth, P. (2008). How young children learn to program with Sensor, Action, and Logic Blocks. *The Journal of the Learning Sciences*, 17(4), 517–550. <https://doi.org/10.1080/10508400802395069>
- Yang, W., Ng, D. T. K., & Su, J. (2023). The impact of story-inspired programming on preschool children’s computational thinking: A multi-group experiment. *Thinking Skills and Creativity*, 47, 101218. <https://doi.org/10.1016/j.tsc.2022.101218>
- Yildiz Durak, H. (2020). The Effects of using different tools in programming teaching of secondary School students on Engagement, computational thinking and reflective thinking skills for Problem solving. *Technology Knowledge and Learning*, 25(1), 179–195. <https://doi.org/10.1007/s10758-018-9391-y>
- Yürük, T. (2007). John Dewey, how we think, a restatement of the relation of reflective thinking to the educative process. *Ankara Üniversitesi İlahiyat Fakültesi Dergisi*, 48(1), 185–188. [https://doi.org/10.1501/ilhfak\\_00000000937](https://doi.org/10.1501/ilhfak_00000000937)



Zhong, B., Xia, L., & Su, S. (2022). Effects of programming tools with different degrees of embodiment on learning Boolean operations. *Education and Information Technologies*, 27(5), 6211–6231. <https://doi.org/10.1007/s10639-021-10884-7>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

## Authors and Affiliations

Xin Gong<sup>1</sup>  · Shufan Yu<sup>2</sup> · Jie Xu<sup>3</sup> · Ailing Qiao<sup>1</sup> · Han Han<sup>4</sup>

✉ Ailing Qiao  
Qiaol@126.com

Xin Gong  
Gongxinjyjs@163.com

Jie Xu  
xjhbsfdx@163.com

Han Han  
hanhan00620@163.com

<sup>1</sup> College of Education, Capital Normal University, 105 West Third Ring North Road, Haidian District, Beijing 100048, China

<sup>2</sup> School of Educational Information Technology, Faculty of Artificial Intelligence in Education, Central China Normal University, Wuhan, Hubei 430079, China

<sup>3</sup> College of Education, Zhejiang University, 866 Yuhangtang Road, Xihu District, Zijingang Campus, Hangzhou 310058, China

<sup>4</sup> Faculty of Education, Beijing Normal University, 19 Xijiekou Outer Street, Haidian District, Beijing 100875, China